НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО

Факультет інформатики та обчислювальної техніки
(назва факультету, інституту)

Кафедра автоматизованих систем обробки інформації і управління
(назва кафедри)

"На правах рукопису"
УДК _____

«До захисту допущено»
Завідувач кафедри

_____ О.А.Павлов
(підпис) (ініціали, прізвище)

“____” _____ 20*18* р.

МАГІСТЕРСЬКА ДИСЕРТАЦІЯ

на здобуття ступеня магістра

за спеціальністю *122 Комп'ютерні науки та інформаційні технології*
(код та назва спеціальності)

спеціалізацією *Інформаційні управляючі системи та технології*
(код та назва спеціалізації)

на тему: *«Задача командного спортивного орієнтування з урахуванням часових вікон»*

Виконала: студентка *VI* курсу групи _____
(шифр групи)

Прохорова Катерина Сергіївна _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник *проф., д.т.н., с.н.с. Гуляницький Л.Ф.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант *к.т.н., доц. Жданова О.Г.* _____
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. ІГОРЯ СІКОРСЬКОГО»

Факультет *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації та управління*
(повна назва)

Рівень вищої освіти *другий (магістерський)*

Спеціальність *122 Комп'ютерні науки та інформаційні технології*
(код і назва)

Спеціалізація *Інформаційні управляючі системи та технології*
(код і назва)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *І.П. Муха*
(підпис) (ініціали, прізвище)
« *12* » *лютого* 2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Прохоровій Катерині Сергіївні
(прізвище, ім'я, по батькові)

1. Тема дисертації *Задача командного спортивного орієнтування з урахуванням часових вікон*

науковий керівник дисертації *Гуляницький Леонід Федорович, д.т.н., с.н.с.*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від " *12* " *березня* 20*18* р. № *885-с*

2. Строк подання студентом дисертації " *5* " *травня* 20*18* р.

3. Об'єкт дослідження *процес складання туристичних маршрутів*

4. Предмет дослідження *задача командного спортивного орієнтування з урахуванням часових вікон*

5. Перелік завдань, які потрібно розробити *розробити метаевристичні алгоритми розв'язання задачі командного спортивного орієнтування з часовими вікнами, дослідити ефективність реалізованих алгоритмів шляхом проведення обчислювального експерименту*

6. Орієнтовний перелік ілюстративного матеріалу *змістовна постановка задачі, схема структурна класів, екранні форми, блок-схема повторюваного локального, блок-схема модифікованого алгоритму, порівняння результатів роботи паралельного і послідовного алгоритмів, результати роботи модифікованого алгоритму повторюваного локального пошуку*

7. Орієнтовний перелік публікацій *«Метод розв'язання задачі командного спортивного орієнтування з часовими вікнами» - к. «Актуальні питання сьогодення», «Математична постановка та огляд методів розв'язування задачі побудови туристичних маршрутів» - к. «Інформатика та обчислювальна техніка - ІОТ-2017»*

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання "*12*" *лютого* 20*18* р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	*Огляд задач, що формалізують проблему побудови туристичних маршрутів*	*15.02.2018*	
2	*Огляд методів розв'язування задачі командного спортивного орієнтування з уракуванням часових вікон*	*19.02.2018*	
3	*Постановка та формалізація математичної моделі задачі*	*26.02.2018*	
4	*Модифікація існуючих методів розв'язання задачі*	*19.03.2018*	
5	*Розробка інформаційного та програмного забезпечення*	*09.04.2018*	
7	*Проведення експериментальних досліджень розроблених алгоритмів*	*16.04.2018*	
8	*Оформлення документації*	*22.04.2018*	
9	*Подання роботи на попередній захист*	*23.04.2018*	
10	*Подання роботи на основний захист*	*05.05.2018*	

Студент _____ *К.С. Прохорова*
(підпис) (ініціали, прізвище)

Науковий керівник _____ *Л.Ф. Гуляницький*
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Магістерська дисертація: 110 с., 20 рис., 22 табл., 1 додаток, 84 джерела.

Актуальність. Всесвітня туристська організація (World Tourism Organization, UNWTO) визначає впровадження нововведень в туризмі однією з основних функцій туристичного маркетингу. Тому, використання інформаційних технологій з метою розвитку туризму, є актуальною задачею на сьогоднішній день. В зв'язку з цим, широкого розповсюдження набули персоналізовані електронні туристичні путівники (Personalized Electronic Tourist guides, PETs), до функціональності яких відноситься задача побудови туристичних маршрутів. При її розв'язанні математична модель може відрізнятись з огляду на те, які умови предметної області враховуються. В даній роботі математичною моделлю виступає задача Командного спортивного орієнтування з часовими вікнами (Team Orienteering Problem with Time Windows, TOPTW).

Оскільки час реагування для програмного забезпечення є важливою ознакою, розробка ефективного алгоритму поставленої задачі на сьогоднішній день є актуальною задачею. Тому, дана робота присвячена дослідженню та удосконаленню розв'язування TOPTW.

Мета роботи і задачі дослідження. Метою є максимізація сумарної корисності побудованих туристичних маршрутів заданої тривалості з врахуванням часових періодів відвідування туристичних місць. Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз відомих результатів розв'язування задачі TOPTW;

– розробити метод (модифікацію існуючого методу) розв'язання задачі з використанням технологій паралельного програмування;

– розробити алгоритмічне забезпечення задачі TOPTW;

– розробити програмну реалізацію алгоритму(ів);

– провести дослідження ефективності розробленого алгоритмічного забезпечення.

Об'єкт дослідження – процес складання туристичних маршрутів.

Предмет дослідження – задача командного спортивного орієнтування з часовими вікнами.

Методи дослідження, застосовані в роботі, базуються на метаевристичних алгоритмах.

Наукова новизна одержаних результатів полягає у модифікації алгоритму повторюваного локального пошуку, порівнянні його з алгоритмом іматійного відпалу, використанні технологій паралельного програмування для модифікації алгоритмів повторюваного локального пошуку і алгоритму імітаційного відпалу для розв'язання задачі задачі TOPTW.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалась у філії кафедри автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» в рамках науково-дослідної теми Інституту кібернетики ім. В. М. Глушкова НАН України: «Розробити математичний апарат, орієнтований на створення інтелектуальних інформаційних технологій розв'язування проблем комбінаторної оптимізації та інформаційної безпеки» (шифр теми: ВФ.180.11).

Публікації. Результати роботи опубліковані у матеріалах науково практичної конференції «Інформатика та обчислювальна техніка-IOT-2017» [82], міжнародної науково-практичної конференції «Актуальні питання сьогодення» [83].

ДЕТЕРМІНОВАНИЙ ЛОКАЛЬНИЙ ПОШУК, ПОВТОРЮВАНИЙ ЛОКАЛЬНИЙ ПОШУК, АЛГОРИТМ ІМІТАЦІЙНОГО ВІДПАЛУ, ЗАДАЧА ПОБУДОВИ ТУРИСТИЧНИХ МАРШРУТІВ, ЗАДАЧА КОМАНДНОГО СПОРТИВНОГО ОРІЄНТУВАННЯ З ЧАСОВИМИ ВІКНАМИ, ПАРАЛЕЛЬНЕ ПРОГРАМУВАННЯ, МЕТАЕВРИСТИЧНІ АЛГОРИТМИ

ABSTRACT

Master dissertation:110 p., 20 fig., 22 tab., 1 appendix, 84 sources.

Relevance. The World Tourism Organization (UNWTO) identifies the introduction of tourism innovations as one of the main functions of tourism marketing. Therefore, the use of information technology for the development of tourism is an actual task to date. Because of that, personalized Electronic Tourist Guides (PETs), which encapsulate tourist trip design problem (TTDP), have become widespread. When TTDP is solved, the mathematical model may differ in terms of what the terms of the subject area are taken into account. In this paper, the problem of Team Orienteering Problem with Time Windows (TOPTW) is the mathematical model.

As the response time for the software is an important feature, developing an effective algorithm for the task to date is an actual task. Therefore, this work is devoted to the research and improvement of the solution methods of TOPTW.

Purpose and objectives of the study. The purpose is to maximize the total usefulness of the built tourist routes of a given duration, taking into account the time periods of visiting tourist places. To achieve this goal it is necessary to solve the following tasks:

- to analyze known results of solving TOPTW;
- to develop a method (modification of the existing method) for solving a problem using parallel programming technologies;
- to develop algorithms for TOPTW;
- to develop software implementation of the algorithms;
- to study the effectiveness of the developed algorithms.

The object of study – the process of designing tourist routes.

Purpose of the study –team orienteering problem with time windows.

The scientific novelty of the obtained results is in modifying the Iterated Local Search algorithm, comparing it with the algorithm of Simulated annealing, using parallel programming technologies for modifying the Iterated Local Search algorithms and the Simulated annealing algorithm for solving TOPTW.

Relationship of work with scientific programs, plans, themes. The work has been carried out at the branch of the Department of Computer-Aided Management and Data Processing Systems of The National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" within the framework of the research topic of the Glushkov Institute of Cybernetics of National Academy of Sciences of Ukraine: "To develop a mathematical apparatus focused on the creation of intelligent information technologies for solving combinatorial optimization and information security problems" (topic code: VF.180.11).

Publications. The results of the work are published in [82, 83].

DETERMINISTIC LOCAL SEARCH, ITERATED LOCAL SEARCH, SIMULATED ANNEALING ALGORITHM, TOURIST TRIP DESIGN PROBLEM, TEAM ORIENTEERING PROBLEM WITH TIME WINDOWS, PARALLEL PROGRAMMING, METAHEURISTIC ALGORITHMS

ЗМІСТ

ВСТУП 10
1 ОГЛЯД ЛІТЕРАТУРИ ЗА ТЕМОЮ ДИСЕРТАЦІЇ 13
1.1 Задача складання туристичних маршрутів 13
1.2 Огляд постановок задачі TTDP 17
1.2.1 Задача Chinese postman problem 17
1.2.2 Задача Rural postman problem 18
1.2.3 Задача Capacitated arc routing problem 18
1.2.4 Задача Maximum benefit chinese postman problem 18
1.2.5 Задача Prize-Collecting Rural Postman Problem 19
1.2.6 Orienteering Problem 19
1.2.7 Arc Orienteering Problem 20
1.2.8 Задача маршрутизації транспортниз засобів 20
1.2.9 Team Orienteering Problem 21
1.2.10 Team Orienteering Arc Routing Problem 22
1.2.11 Team orienteering problem with time windows 22
1.2.12 Time-dependent Team orienteering problem 22
1.2.13 Time-dependent Team Orienteering Problem with Time Window 23
1.2.14 Mixed Team Orienteering Problem with Time Windows 23
1.2.15 The Chinese Postman Problem with load-dependent costs 23
1.2.16 The Stacker Crane Problem 24
1.3 Огляд існуючих методів розв’язання задачі TOPTW 25
1.4 Паралельні обчислення – перспективний напрямок в розробці алгоритмічних методів для TTDP 27
1.5 Практичні застосування задачі TOPTW 28

1.6 Постановка задачі дослідження 29
1.7 Висновки до розділу 30
2 РОЗВЯ'ЗУВАННЯ ЗАДАЧІ TOPTW 31
2.1 Змістовна постановка задачі 31
2.2 Математична постановка задачі 31
2.3 Алгоритм детермінованого локального пошуку 34
2.3.1 Ключові аспекти реалізації детермінованого локального пошуку 35
2.4 Специфіка детермінованого локального пошуку для TOPTW 38
2.5 Метод повторюваного локального пошуку 38
2.5.1 Загальна схема алгоритму ILS 39
2.6 Специфіка методу повторюваного локального пошуку для TOPTW 41
2.6.1 Побудова початкового розв'язку 42
2.6.2 Алгоритм вставки нової вершини 43
2.6.3 Збурення розв'язку 44
2.7 Модифікація алгоритму ILS 45
2.8 Паралельне обчислення околу 45
2.9 Приклад розв'язання задачі 45
2.10 Алгоритм імітаційного відпалу 53
2.11 Специфіка алгоритму імітаційного відпалу задачі TOPTW 57
2.12 Висновки до розділу 57
3 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 58
3.1 Вхідні данні 58
3.2 Вихідні данні 59
3.3 Засоби розробки 60
3.4 Архітектура програмного забезпечення 64

3.4.1 Опис класів 64
3.4.2 Специфікація функцій 66
3.5 Висновки до розділу 75
4 ТЕХНОЛОГІЧНИЙ РОЗДІЛ 76
4.1 Вимоги до технічного забезпечення 76
4.2 Керівництво користувача 76
4.3 Висновки до розділу 80
5 РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ 82
5.1 Дослідження ефективності використаних алгоритмів 82
5.2 Аналіз отриманих результатів 91
5.3 Висновки до розділу 93
ЗАГАЛЬНІ ВИСНОВКИ 94
ПЕРЕЛІК ПОСИЛАНЬ 95
ДОДАТОК А ГРАФІЧНИЙ МАТЕРІАЛ 103
ПЛАКАТ 1 ЗМІСТОВНА ПОСТАНОВКА ЗАДАЧІ 104
ПЛАКАТ 2 БЛОК-СХЕМА ПОСЛІДОВНОГО АЛГОРИТМУ ПОВТОРЮВАНОГО ЛОКАЛЬНОГО ПОШУКУ 105
ПЛАКАТ 3 БЛОК-СХЕМА МОДИФІКОВАНОГО АЛГОРИТМУ ПОВТОРЮВАНОГО ЛОКАЛЬНОГО ПОШУКУ 106
ПЛАКАТ 4 СХЕМА СТРУКТУРНА КЛАСІВ БІБЛІОТЕКИ TOPTWLib 107
ПЛАКАТ 5 РЕЗУЛЬТАТИ РОБОТИ МОДИФІКОВАНОГО ПОВТОРЮВАНОГО ЛОКАЛЬНОГО ПОШУКУ 108
ПЛАКАТ 6 ПОРІВНЯННЯ РЕЗУЛЬТАТІВ РОБОТИ ПАРАЛЕЛЬНОГО І ПОСЛІДОВНОГО АЛГОРИТМІВ 109
ПЛАКАТ 7 ЕКРАННІ ФОРМИ 110

ВСТУП

Туризм в Україні є важливою і перспективною галуззю економіки. Україна посідає одне з провідних місць в Європі за кількістю об'єктів історико-культурної спадщини, що викликають інтерес у вітчизняних та іноземних туристів. Щороку Україну відвідують понад 20 мільйонів туристів. Наразі укладено 12 міжнародних угод про співпрацю в галузі туризму з державами ЄС. Україна оголосила 2017 рік – роком туризму в контексті року сталого розвитку туризму, заснованого Генасамблеєю ООН. На сьогоднішній день у туристичному секторі України працює 10% населення. Туризм перетинається з діяльністю 40 галузей української економіки та входить до переліку п'яти галузей України, що приносять найвищі доходи. У минулому році Україну відвідали понад 13,3 млн іноземних громадян, що на 6,7% більше, ніж у 2015 році. За перше півріччя 2017 року в Україні побували 6,3 млн іноземців. За аналогічний період минулого року нашу країну відвідали 5,8 млн іноземців [80].

Проте, формування привабливого туристичного іміджу країни ускладнюється політичними, соціально-економічними факторами, відсутністю достовірної та актуальної інформації щодо туристичних ресурсів та послуг.

Туризм, як і всі інші форми господарської діяльності, потребує використання принципів маркетингу. Згідно з французькими науковцями Ланкаром та Ольє, туристичний маркетинг являє собою серію методів і прийомів, вироблених для дослідження, аналізу та вирішення поставлених завдань щодо найповнішого задоволення потреб туристів, а також визначення економічно раціональних способів ведення справ в галузі туризму.

Основним фокусом туристичного маркетингу згідно даних Всесвітньої організації туризму (World Tourism Organization, UNWTO) є впровадження нововведень в туризм [69]. Тому, використання інформаційних технологій з метою розвитку туризму, є актуальною задачею на сьогоднішній день. В зв'язку з цим, розроблється велика кількість мобільних і Web-застосувань для туризму. Наприклад, широкого розповсюдження набули персоналізовані електронні

туристичні путівники (Personalized Electronic Tourist guides, PETs), до функціональності яких входить побудова туристичних маршрутів.

Дана функціональність сприяє розповсюдженню актуальної інформації щодо туристичних ресурсів та послуг у доступному всім туристам форматі. Вона може бути корисна як для іноземних туристів, так і для громадян України. Внутрішній туризм набуває обертів, тому необхідність у застосуваннях для туризму в Україні зростатиме. Як повідомляє Держстат, у минулому році кількість внутрішніх туристів в Україні становила понад 453,5 тис. осіб.

PETs дозволяють туристам будувати маршрути на невідомій місцевості враховуючи їх побажання. При їх створені виникає проблема побудови туристичних маршрутів. Процес побудови цих маршрутів є об’єктом дослідження роботи. Формально описати дану проблему можна в термінах задачі командного спортивного орієнтування з часовими вікнами, що є предметом дослідження. Метою є максимізація сумарної корисності побудованих туристичних маршрутів заданої тривалості з врахуванням часових періодів відвідування туристичних місць. Для досягнення поставленої мети необхідно вирішити такі задачі:

- провести аналіз відомих результатів розв’язування задачі TOPTW;
- розробити метод (модифікацію існуючого методу) розв’язання задачі з використанням технологій паралельного програмування;
- розробити алгоритмічне забезпечення задачі TOPTW;
- розробити програмну реалізацію алгоритму(ів);
- провести дослідження ефективності розробленого алгоритмічного забезпечення.

З огляду на складність задачі TOPTW методами дослідження обрані метаевристичні алгоритми: повторюваний локальний пошук і метод імітаційного відпалу.

Науковою новизною в є використання технологій паралельного програмування для зменшення часу роботи алгоритмів, аналіз роботи імплементованих алгортмів, імплементація модифікації повторюваного локального пошуку. Дослідження проведені у напрямку пошуку ефективних методів

розв’язування задачі TOPTW необхідні для конструювання якісного алгоритмічного забезпечення для PETs. Тому, дана робота присвячена дослідженню та удосконаленню розв’язування задачі TOPTW.

1 ОГЛЯД ЛІТЕРАТУРИ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

1.1 Задача складання туристичних маршрутів

Задача складання туристичних маршрутів (tourist trip design problem, TTDP) була сформульована у [70]. Дана задача ставить на меті сформувати туристичні маршрути на один чи декілька днів, що обмежені у часі. Маршрути повинні максимізувати сатисфакцію від відвідування туристичних об‘єктів. Дана задача виникла як спосіб автоматизації процесу складання туристичних маршрутів у реальному часі, оскільки на час написання роботи (2007 рік) більшість користувалася газетами, журналами, інтернет виданнями про туризм для планування поїздок. Тому автори запропунували концепцію мобільного застосування, що розв‘язує дану проблему. В їх роботі математичною постановкою служить проблема спортивного орієнтування (Orienteering Problem, OP). OP являє собою задачу з області Дослідження операцій [64]. В такому формулюванні TTDP на виході має один маршрут. Подальшу роботу над даною проблемою вони опублікували у [59]. Детальніше розглянуто проблему визначення корисності туристичних об’єктів за допомогою векторної моделі [13] і запропоновано метод розв‘язання OP – керований локальний пошук [6].

Необхідно зазначити, що формулювання TTDP у термінах OP обмежує можливості користувачів застосування, що описане у вищерозглянутих роботах, адже, зазвичай тури включають декілька днів, а математична постановка у термінах OP дозволяє лише один маршрут. У подальших роботах дослідників, що працюють над тематикою, представлені інші підходи до формування математичної постановки для TTDP з одним маршрутом. Необхідно зазначити, що вагома частина пізніших робіт присвячена TTDP з багатьма маршрутами.

Обширний огляд з даної проблеми проведено у [5]. Розглянуто такі аспекти, що пов‘язані з TTDP:

– постановка задачі TTDP і її варіації, що виникають від того, в якій мірі враховуються реальні умови предметної області;

– основний функціонал систем, що розв‘язують TTDP;

– математичні постановки, в термінах яких може бути представлена TTDP;

– перспективні напрямки досліджень.

– До додаткової функціональності можна віднести:

– візуалізація на карті;

– навігація.

У літературі можна знайти різні назви для застосувань (здебільшого мобільних застосувань), що вирішують проблему TTDP. Наприклад, у [5] застосовується назва «Персоналізовані електронні путівники» (Personalized Electronic Tourist guides, PETs). Згідно з [58], PETs включають таку основну функціональність:

– генерування списку туристичних об’єктів;

– складання туристичних маршрутів;

– функціональність для редагування згенерованих туристичних маршрутів.

За генерування списку туристичних об’єктів відповідає рекомендаційна система, а складання туристичних маршрутів – це алгоритм, який на основі даних, що генеруються рекомендаційною системою, формує розв‘язок - туристичний(і) маршрут(и). Маршрут являє собою послідовність туристичних об’єктів. Він обмежений в часі. Оптимальне рішення являє собою маршрут максимальної корисності (сатисфакції).

В багатьох публікаціях підсистеми генерування списку туристичних об’єктів і складання туристичних маршрутів розглядаються в комплексі, бо перша генерує вхідні дані для другої. Огляд відносно рекомендаційних систем для TTDP наведено у [31].

В літературі для TTDP наведено ряд задач в рамках яких наведено математичну постановку. Це спричинено тим, що при розв‘язанні проблеми врахувати всі обмеження предметної області практично неможливо, оскільки їх надзвичайно багато, наприклад, деякі з них:

– різний час роботи туристичних об’єктів;

– різний час на рух у різних напрямках між двома пунктами;

– різний час на проходження однієї ділянки, в залежності від часу доби.

Крім того, туристичні об’єкти можуть бути представлені як видовими маршрутами, так і окремими пунктами; кількість маршрутів може бути різною. Тому дослідники, зазвичай, обирають декілька аспектів, на яких вони концентруються. Проте, мінімальний набір вхідних даних для будь-якої математичної постановки такий:

– набір ТО. Для кожного ТО можу бути надані певні атрибути (місцезнаходження, робочі години, тип і т. д.);

– час на переміщення між ТО;

– корисність кожного ТО, що генерується рекомендаційною системою;

– кількість маршрутів, що повинна бути згенерована;

– час на відвідування ТО;

– час відведений для кожного маршруту.

Перелік задач в рамках яких наводиться математична постановка задачі TTDP наведено в таблиці 1.1.

Таблиця 1.1 – Математичні постановки TTDP

Назва	Абревіатура	Опис
Chinese postman problem	CPP	Обхід всіх ребер графа
Rural postman problem	RPP	Обхід певних ребер графа
Capacitated arc routing problem	CARP	Обхід ребер декількома транспортними засобами
Maximum benefit chinese postman problem	MBCPP	Обхід ребер графа з метою максимізації корисності від відвідування ребер
Prize-Collecting Rural Postman Problem	PCRPP	Побудова маршруту максимальної корисності, по ребрам, щоб разово отримати користь від їх відвудування
Maximum benefit chinese postman problem	MBCPP	Обхід ребер графа з метою максимізації корисності від відвідування ребер
Prize-Collecting Rural Postman Problem	PCRPP	Побудова маршруту максимальної корисності, по ребрам, щоб разово отримати користь від їх відвудування.

Продовження таблиці 1.1

Назва	Абревіатура	Опис
Orienteering Problem	OP	Маршрут обмежений у часі максимальної корисності корисності, коли корисність задана для вершин
Vehicle routing problem	VRP	Маршрути мінімальної вартості, що задовольняють нормам, встановленим для кожної вершини
Team Orienteering Problem	TOP	Маршрут обмежений у часі максимальної корисності, коли корисність задана для вершин
Arc Orienteering Problem	AOP	Маршрут обмежений у часі максимальної корисності, коли корисність задана для ребер
Team Orienteering Arc Routing Problem	TOARP	Маршрути обмежений у часі максимальної корисності
Team Orienteering Problem with Time Windows	TOPTW	Маршрути максимальної корисності, що обмежені у часі, коли для вершин задані часові вікна і корисність
Mixed Team Orienteering Problem with Time Windows	MTOPTW	Маршрути обмежений у часі максимальної корисності коли для певних вершин і ребер, що включені і мають корисність, задані часові вікна
Mixed Team Orienteering Problem with Time Windows	MTOPTW	Маршрути обмежений у часі максимальної корисності коли для певних вершин і ребер, що включені і мають корисність, задані часові вікна
The Chinese Postman Problem with load-dependent costs	CPP-LC	Обхід всіх ребер графа, де вартість відповідних ребер змінна
Time-dependent Team orienteering problem	TDTOP	Маршрути максимальної корисності, що обмежені у часі, коли для вершин задана корисність. Час проходження по ребрах залежить від години доби.

Продовження таблиці 1.1

Назва	Абревіатура	Опис
Time-dependent Team Orienteering Problem with Time Window	TDTOPTW	Маршрути максимальної корисності, що обмежені у часі, коли для вершин задані часові вікна і корисність. Час проходження по ребрах залежить від години доби.
The Stacker Crane Problem	SCP	Маршрут на змішаному графі мфнімальної вартості, що включає всі дуги

1.2 Огляд постановок задачі TTDP

1.2.1 Задача Chinese postman problem

Відомою задачею з класу ARP є Задача (китайського) листоноші (Chinese postman problem або Route inspection problem) [46] метою якої є пошук пошук найкоротшого циклу в графі, що включає всі ребра. Існують варіанти задачі для орієнтованих та неорієнтованих графів та для графів, частина ребер яких орієнтована, а частина — ні.

Задача отримала назву завдяки Алану Голдману з NIST, що назвав її так, бо першим дослідником цієї задачі був китайський математик Мей-Ку Кван.

Задача визначена на графі $G = (V, E \cup A)$, де V являє собою множину вершин, а E – множину ребер, а A - дуг. Граф G називається орієнтовним, якщо множина E порожня, неорієнтованим, якщо A порожня, і змішаним, якщо множини E і A не порожні. В відповідності до цього розрізняють орієнтовану, неорієнтовану і змішану задачі китайського листоноші. Узагальненням цих трьох задач є Windy Postman Problem. Вона визначена на ненаправленому графі, де ребро може мати різну вартість обходу у різних напрямках.

Мета: знаючи вартість обходу c_{ij} ребер або/та дуг (v_i, v_j) необхідно побудувати маршрут найменшої вартості T представлений вектором виду $(v_1, v_2, \dots, v_n)$, де (v_i, v_{i+1}) належить $E \cup A$ при $i = 1, \dots, n-1$ та $v_n = v_1$. Даний маршрут повинен включати всі ребра з $E \cup A$.

1.2.2 Задача Rural postman problem

Задача сільського листоноші (Rural postman problem) є узагальненням CPP. Задано множину $S \subset E \cup A$ і необхідно побудувати замкнений маршрут мінімальної вартості через ребра, що містяться у S. Щоб звести RPP до CPP треба множину S визначити такою, що вона включає всі ребра з $E \cup A$.

Мета: знаючи вартість обходу c_{ij} ребер або/та дуг (v_i, v_j) необхідно побудувати маршрут найменшої вартості T представлений вектором виду $(v_1, v_2, \dots, v_n)$, де (v_i, v_{i+1}) належить $E \cup A$ при $i = 1, \dots, n-1$ та $v_n = v_1$. Даний маршрут повинен включати всі ребра з S.

1.2.3 Задача Capacitated arc routing problem

Задача Capacitated arc routing problem є розширенням задачі RPP. Вводяться додаткові обмежень пропускної здатності на ребрах (дугах). Є m однакових транспортних засобів, кожен з яких має ємність Q. Задається вершина графу що являє собою депо, і кожне необхідне ребро має невід'ємний попит. Маршрут допустимий, якщо він містить депо і якщо сумарний попит не перевищує місткість Q. Число m транспортних засобів може бути задане апріорі або може бути змінною для якої приймається рішення.

Мета: знайти множину допустимих маршрутів для транспортних засобів, що мають сумарну мінімальну вартість, а кожне необхідне ребро обслуговується рівно одним транспортним засобом [9].

1.2.4 Задача Maximum benefit chinese postman problem

Вперше дана задача була сформульована Маландракі і Даскіном [46] у 1993 році і визначена на неорієнтовному графі.

Для кожного ребра $e \in E \cup A$ визначено прибуток від проходження p_e^i $i = 1,2 \dots n_e$. Цю величину можна трактувати як потребу у сервісі певного ребра. З кожним проходом і по е прибуток p_e^i зменшується, а після того, як через ребро е пройдено n_e разів, він дорівнює нулю. Це означає, що дане ребро більше не

потребує сервісу і подальші проходження по цьому ребру можна вважати збитковими. Витрати $c_e^i\, i = 1,2 \ldots n_e$ на проходження по ребру e незмінні, а коли по ребру пройдено n_e разів, витрати для проходження цього ребра зростають. Результуюча корисність проходження по певному ребру визначається як $p_e^k - c_e^k$, де k позначає скільки разів ребро e зустрічається у маршруті.

Мета: знайти маршрут для якого результуюча сумарна корисність ребер буде максимальною.

1.2.5 Задача Prize-Collecting Rural Postman Problem

Задача була сформульована вперше Араосом, Фернандезом і Золтаном у 2006 році. Вона являє собою частковий випадок задачі MBCPP, коли кожне прибуток p_e^i з ребра можна отримати лише при першому проходженні, тобто $n_e = 1$ [11].

Мета: знайти цикл через депо для якого результуюча сумарна корисність ребер (дуг) буде максимальною.

1.2.6 Orienteering Problem

Задача вперше була представлена у [37]. Вона отримала свою назву, бо її математична постановка якнайкраще моделює гру спортивного орієнтування. Суть гри в тому, що спортсмени за обмежений час відвідують пункти в певній місцевості. Всі вони починають і повинні закінчити в наперед визначених пунктах. За відвідування інших пунктів нараховуються бали, причому, кількість балів різна для кожного пункту. Перемагає той спортсмен, маршрут якого матиме найбільшу сумарну кількість балів.

В літературі дана задача зустрічається і під іншими назвами: Selective Traveling Salesperson Problem (STSP) [38], Maximum Collection Problem (MCP) [39] and Bank Robber Problem (BRP) [12,40].

Задано початкову і кінцеву вершини. Для кожної вершини визначено корисність c_{ij} від проходження по ній. Більш того, лише перше проходження

приносить корисність. Задано величину t_{ij} для дуг (i, j), що зазвичай трактується як час потрібний на проходження дуги. На час проходження для всього побудованого маршруту задане обмеження величиною T.

Мета: знайти цикл через початкову вершину максимальної прибутковості, щоб час проходження ним не перевищував T.

1.2.7 Arc Orienteering Problem

Дана задача визначена на направленому графі [57]. Задано початкову вершину з якої треба побудувати маршрут. Для кожної дуги визначено корисність c_{ij} від проходження по ній. Більш того, лише перше проходження дугою приносить корисність. Задано величину t_{ij} для дуг (i, j), що зазвичай трактується як час потрібний на проходження дуги. На час проходження для всього побудованого маршруту задане обмеження величиною T.

Мета: знайти цикл через початкову вершину максимальної прибутковості, щоб час проходження ним не перевищував T.

1.2.8 Задача маршрутизації транспортниз засобів

Формально класичну задачу маршрутизації ТЗ (*Vehicle routing problem*) можна представити так: дано граф , де множина вершин визначає клієнтів, яких треба відвідати, і депо, з якого починаються і яким закінчуються всі маршрути [34]. Кожен клієнт характеризується певним попитом . Дуги, що поєднують вершини зважені. Вагою дуги є час переїзду ТЗ від клієнта до клієнта , цей час включає в себе час обслуговування клієнта. Кожен клієнт обслуговується тільки одним транспортним засобом і тільки один раз. Транспортний засіб не може обслужити більше клієнтів, ніж дозволяє його вантажопідйомність. Кожен автомобіль покидає депо і прибуває туди один раз. На рисунку 1.1 схематично зображено можливий розв’язок задачі.

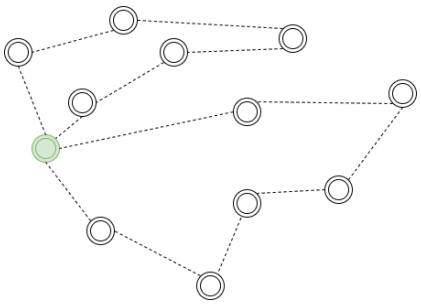


Рисунок1.1 – Схематичне зображення можливого розв’язку задачі VRP

Мета: побудувати маршрути, що являють собою розбиття множини вершин графу, такі, що сумарний витрачений час мінімальний, а всі потреби клієнтів задоволено.

1.2.9 Team Orienteering Problem

Задача спортивного командного орієнтування (TOP) була вперше представлена у [50]. Вона являє собою розширення задачі OP до випадку з багатьма маршрутами. В літературі її можна також знайти під назвою Multiple Tour Maximum Collection Problem (MTMCP) [20].

На рисунку 1.2 схематично зображено можливий розв’язок задачі, коли необхідно побудувати 2 маршрути. Початкова вершина зеленого кольору.Той факт, що не всі вершини включено, є основною відмінністю даної задачі від VRP.

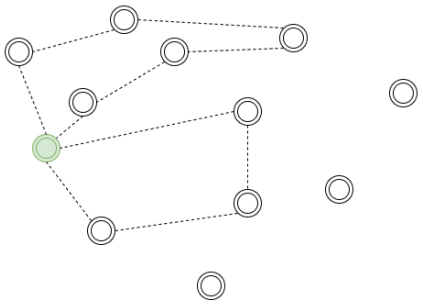


Рисунок 1.2 – Схематичне зображення можливого розв’язку задачі TOP

Мета: знайти k початкову вершину максимальної прибутковості, щоб час проходження ним не перевищував T.

1.2.10 Team Orienteering Arc Routing Problem

Задача Team Orienteering Arc Routing Problem являє собою розширення AOP. Задано декілька транспортних засобів для яких потрібно побудувати маршрути, що не перетинаються один з одним.

Одним із застосувань TOARP є транспортування вантажними автомобілями. Деякі клієнти повинні обслуговуватися, тоді як інші можуть бути відкладені або взагалі не обслуговуватися. Наприклад, обслуговування найменш прибуткових клієнтів може бути передане на аутсорсинг.

Мета: Побудувати набір маршрутів таких, що час на проходження кожного не перевищує T і всі обов'язкові дуги пройдено. Сумарна корисність по всім маршрутам повинна бути максимальною [2].

1.2.11 Team orienteering problem with time windows

Дана задача є розширенням TORP, у тому сенсі, що кожна дуга має час візиту і час відвідування. Таким чином, прибуток з відвідування дуги можна отримати лише в певному часовому вікні. Вперше дана задача була представлена у [70].

Мета: Побудувати набір маршрутів, таких, що кожен з них починається і завершується в одній вершині, час на проходження кожного не перевищує T. Сумарна корисність по всім маршрутам повинна бути максимальною.

1.2.12 Time-dependent Team orienteering problem

Дана проблема є розширенням TOP. Задача ускладнюється тим, що в різні проміжки часу проходження по певній дузі може бути швидшим або довшим. Наприклад, якщо розглянути випадок коли маршрут частково проходиться пічки, а частково проїжджається громадським транспортом. Знаючи розклад громадського транспорту, можна оцінити скільки займе маршрут. На загальний час маршруту буде впливати скільки займає пройти певні відстані пішки, час очікування громадського транспорту і час переїздів. Якщо пішохід обиратиме маршрут, що дозволить йому підлаштуватися під розклад громадського транспорту, це дозволить скоротити час переміщення між певними ділянками.

Мета: Побудувати набір маршрутів, таких, що кожен з них починається і завершується в одній вершині, час на проходження кожного не перевищує T. Сумарна корисність по всім маршрутам повинна бути максимальною.

1.2.13 Time-dependent Team Orienteering Problem with Time Window

TDTOPTW - це проблема, яка добре моделює складні та реалістичні вимоги TTDP. TDTOPTW є особливо складною, оскільки він додає часові залежності, що присутні у TOPTW. У [7979] описано проблему, яка відноситься до TDTOPTW, і ROSE - мобільний додаток, що допомагає знаходити події та місцеположення, переміщаючись через зв'язки громадського транспорту.

Мета: Побудувати набір маршрутів, таких, що кожен з них починається і завершується в одній вершині, час на проходження кожного не перевищує T. Сумарна корисність по всім маршрутам повинна бути максимальною.

1.2.14 Mixed Team Orienteering Problem with Time Windows

Задача TOAP має аналог для вершин. Умови залишаються незмінними, але корисність накопичується за проходження вершин, а не ребер. Задача має назву Team Orienteering Problem (TOP).

Задача Mixed Team Orienteering Problem with Time Windows є комбінацією TOP і TOAP, причому задані часові вікна. Тобто, корисність задається ребрам і вершинам [28].

Мета: Побудувати набір маршрутів, таких, що час на проходження кожного не перевищує T. Сумарна корисність по всім маршрутам повинна бути максимальною.

1.2.15 The Chinese Postman Problem with load-dependent costs

Задача буле предсталена у 2016 році як розширення CPP [62]. Сформульована вона з метою побудови маршрутів, спланованих таким чином, щоб викиди вуглекислого газу спричинені транспортним засобом зменшувались.

Задача визначена на зв‘язаному неорієнтовному графі. Задана вершина, яка являє собою депо. Для кожного ребра $e \in E$ визначено попит $q_e \geq 0$ і довжину $d_e \geq 0$ ребра. Попит можна трактувати як вагу солі, що необхідно відвантажити на ребрі. Вага транспортного засобу складається з спорядженої маси і маси, що навантажена на транспортний засіб $\sum_{e \in E} q_e$. Коли транспортний засіб проходить по ребру e він відвантажує на ньому вагу q_e. Таким чином, його сумаран вага зменшується. $q_e = 0$ для ребер, що вже були пройдені.

Витрати на проходження ребра визначаються як добуток повної маси транспортного засобу і довжини ребра. Результатом того, що повна маса транспортного засобу змінюється під час проходження по ребрам витрати на проходження ребер не є постійними виличинами.

Мета: Побудувати цикл через депо мінімальної вартості, що охоплює всі ребра $e \in E$.

1.2.16 The Stacker Crane Problem

Задача Stacker Crane Problem [64] отримала свою назву від застосування до практичної проблеми експлуатації крана. Кран, починаючи з вихідної вершини, повинен виконати набір рухів, і повернутися в початкове положення.

SCP була запропонована Фредеріксоном, Хехтом, і Кімом, які описали три версії задачі в залежності від кінцевого положення крана. В першій версії кран повинен повернутися в точку з якої починав. В другій, кран зупиняється на певній вершині, не обов'язково тій, де починав, в той час як для третьої варіації дозволяється закінчити в будь-якій вершині. Перша версія може легко бути приведена до другої.

Задача визначена на мішаному графі $G = (V, E \cup A)$, де V являє собою множину вершин, а E – множину ребер, а A – дуг, множина E не порожня і A не порожня. Кожна ланка (дуга або ребро) графа має пов'язану з ним невід'ємну вартість.

Мета: знайти тур мінімальної вартості, знаючи початкову вершини, такий, що всі елементи множини A включено у маршрут.

1.3 Огляд існуючих методів розв’язання задачі TOPTW

За своєю складністю задачі TOPTW не менш складна ніж TOP. В свою чергу, TOP є розширенням задачі OP (OP є частковим випадком TOP) тому, як і OP, TOP і TOPTW є NP-складними задачами. Доведення того, що OP є NP-складною задачею наведено у [29].

TOPTW може бути оптимально розв’язана точними методами. Втім, такі методи можуть використовуватись лише для розв’язку задач із невеликою кількістю вузлів. До цих методів відносяться динамічне програмування і метод гілок та меж. Наприклад, у [61] TOPTW розв‘язано точними методами для графу з 30 вершинами.

Зважаючи на складність проблеми, переважна більшість літератури присвячена розв‘язанню проблеми неточними методами з застосуванням евристичних та метаевристичних методів.

Евристичні методи – сукупність прийомів в пошуку розв’язання задачі, які дозволяють обмежити перебір, тобто проводиться відносно обмежений пошук по простору рішень, і зазвичай знаходять розв’язок за прийнятний час. Недолік даних методів в тому, що вони є приблизними, розв’язок, отриманий даними методами, може бути далеким від оптимального. Перевага в тому, що вони дозволяють знайти прийнятний розв’язок NP-повних задач великої розмірності за прийнятний час. Евристичні методи часто протиставляються формальним методам розв’язання, які спираються, наприклад, на точні математичні алгоритми.

Метаевристики – загальна назва для будь-якого стохастичного алгоритму оптимізації, який використовується в якості «останньої надії» на шляху до вирішення завдання з використанням випадкового пошуку або повного перебору. Тобто це такий метод вирішення широкого класу обчислювальних завдань шляхом комбінування існуючих процедур з відкритим інтерфейсом і закритою реалізацією, яке призводить до максимально ефективного вирішення. В метаевристичних методах акцент робиться на ретельне вивчення найбільш перспективних частин простору рішень. Якість одержуваних рішень виходить вищою, ніж у отриманих класичними евристиками.

Здебільшого, існуючі методи для розв‘язання TOPTW являють собою метаевристики, що включають:

– крок вставки нового пункту у одному з маршрутів, що ітеративно відбувається, доки не знайдено допустиме рішення;

– крок диверсифікації для уникнення локального оптимуму.

Ці два кроки повторюються, доки критерій виходу не задовольняється. Залежно від критерію вставки, існуючі методи розроблені як детерміновані(ті, що завжди продукують однаковий розв‘язок для певних вхідних даних), чи стохастичні, чи імовірносні(ті, що включають ступінь випадковості в генерацію розв‘язоку). Вважається, що застосування імовірносних методів приводить до генерації розв‘язків високої якості, бо проводиться більш обширний пошук у просторі розв‘язків, за рахунок підвищення часу виконання.

У [65] запропоновано алгоритм локального пошуку TOPTW зі змінними околами. У ході виконання алгоритм намагається замінити сегмент шляху вершинами, що не включені у маршрут, та приносять більшу корисність. Для цього вирішується задача про призначення, що відноситься до TOPTW, і на основі рішення вставляються нові дуги у маршрут.

У [43] запропоновано алгоритм на основі метаевристики імітації відпалу для TOPTW. На кожній ітерації з околу поточного розв‘язку генерується новий шляхом перестановок, інверсій чи вставок з однаковою імовірністю. Отриманий розв‘язок стає новим, якщо він кращий за попередній (корисність більша), в протилежному випадку він приймається з певною імовірністю, що зменшується відповідно втраті корисності. Після певної кількості ітерацій цього алгоритму застосовуються локальний пошук.

У [43] запропоновано алгоритм на основі ітеративного локального пошуку, що застосовує кроки «струсу» і «вставки». Крок вставки відповідальний за додавання послідовності вершин, таким чином, щоб розв‘язок залишався допустимим (кожна вершина повинна відвідуватись у своєму часовому вікні). Для кожної вершини, що може бути вставлена, визначається відношення корисності до затримки у часі, що спричиняє вставка. Обирається вершина з найбільшим значенням даного

відношення. Крок збурення застосовується для уникнення локального оптимуму. На кожному кроці заміняться для кожного маршруту одна чи більше вершин на ті, що не включені в розв‘язок, незважаючи на їх корисність.

Алгоритм мурашиної колонії запропонований для розв‘язку задачі у [72]. Комбінацію жадібного рандомізованого адаптивного пошуку і еволюційного локального пошуку застосовано у [35, 35].

1.4 Паралельні обчислення – перспективний напрямок в розробці алгоритмічних методів для TTDP

Одна з основних задач в розробці алгоритмічних методів для TTDP – це швидка реакція на запити користувача. Паралельні обчислення можуть стати механізмом для досягнення хороших результатів у цьому напрямку. Враховуючи вищерозглянуті методи розв‘язання задачі, евристичні та метаевристичні методи найкраще піддаються для розпаралелювання, оскільки простір рішень надає багато варіацій для паралельних обчислень. Наприклад, згідно [36] можна представити паралельну версію локального пошуку за допомогою розбиття простору рішень на підмножини. Таким чином, можна запустити евристику на кожній підмножині.

Інший варіант – запустити евристику чи різні евристики на загальному просторі рішень починаючи з одного чи різних початкових розв‘язків. В ході роботи потоки можуть обмінюватись результатами відносно кращих знайдених розв‘язків на даний момент. Цікавий аспект цих підходів полягає у тому, що вони також можуть забезпечити генерування нових евристик з якіснішими розв‘язками, оскільки вони можуть шукати в просторі рішення і поєднувати рішення так, що це було б дуже дорого імітувати за послідовної реалізації. Хоча паралельна евристика були запропоновані в літературі для VRP і TSP [23, 25, 26, 56, 4] паралельні рішення для TTDP відсутні (або нечисленні, тому й не охоплені під час огляду літератури), і розробка нової паралельної евристики для TTDP може зменшити час реакції застосувань на запит користувачів.

1.5 Практичні застосування задачі TOPTW

Окрім TTDP, задача TOPTW має інші практичні застосування:

– задача краудсорсингу;

– задача розподілу медичного персоналу для домашніх візитів.

Дані проблеми можна розглянути у рамках задачі маршрутизації транспортних потоків (Vehicle routing problem, VRP), але формулювання у термінах OP дозволяє краще врахувати специфіку предметних областей даних задач.

Задача краудсорсингу, коли існує набір робіт для краудсорсингу і набір працівників, що можуть виконувати різні роботи, була розглянута у [67] як задача класу OP. Зазвичай застосування для краудсорсингу, для формування набору задач роблять відбір на основі територіального положення працівника. Обираються роботи, що лежать в певному радіусі, що охоплює поточне положення працівника. Підходи у [51, 66] направлені на те, щоб запропонувати більш гнучкий підхід до даної проблеми. Пункти, що рекомендуються працівнику обираються на основі його траєкторії і корисності робіт. Оцінка корисність робіт може надаватись залежно від таких факторів як оплата за роботу, навики працівника, вподобання працівника і т.д..

Задача розподілу персоналу була розглянута в літературі для індустрії надання медичних послуг в домашніх умовах. Коли дану задачу сформувати у термінах VRP, то дана вона постає як проблема задоволення попиту клієнтів. Коли задача формується у термінах OP, то вона розглядається з точки зору компанії, що надає послуги. Постановка у термінах OP дозволяє врахувати ситуацію, коли кількість клієнтів така велика, що перевищує можливості персоналу надати медичні послуги. Візитам до пацієнта має певну корисність, щоб максимізувати сумарну сатисфакцію обирається максимальний набір клієнтів, що зможе обслужити компанія за певний проміжок часу.

У [60] представлено задачу призначення медичних візитів в домашніх умовах з врахуванням випадковості у часі обслуговування і переміщення. Було запропоновано методи розв‘язку даної проблеми, що тестувались на синтетичних

наборах даних. Були сформовані набори як для детермінованих, так і стохастичних випадків.

У [30] задачу сформовано у термінах TOP with soft Time Windows and Variable Profit, TOPsTWVP. На меті ставиться максимізація сатисфакції пацієнтів, тому кожен візит має корисність. Передбачається, що візити обмежені у часі, але можливі також і візити з запізненням. Запізнення призводять до зниження корисності візиту. Також враховується, що пацієнти очікують певного лікаря, тому, якщо візит заплановано не з цим лікарем, то корисність знижується.

Треба зазначити, що проблеми класу OP добре підходять для формулювання задач побудови маршрутів для розподілу персоналу у різних предметних областях, не лише для області надання медичних послуг.

Якщо сформувати дану задачу у термінах TOPTW, то можна врахувати такі фактори:

– попит на послуги більший за пропозицію, тому не всі клієнти можуть бути обслуговані за виділений час;

– обслуговування відбувається у рамах певних часових вікон;

– щоб надати послуги наступному клієнту необхідно витратити час на переміщення до нього;

– обслуговування клієнта займає певний час;

– обслуговування клієнтів може мати різну корисність;

1.6 Постановка задачі дослідження

Мета дисертаційної роботи – максимізація сумарної корисності набору обмежених у часі туристичних маршрутів із урахуванням часових вікон.

Для досягнення поставленої мети, необхідно вирішити такі **задачі**:

- провести аналіз відомих результатів у розв’язанні задачі TOPTW;
- розробити метод розв’язання задачі з застосуванням паралельних обчислень;
- розробити алгоритмічне забезпечення задачі;
- розробити програмну реалізацію алгоритму;

- провести дослідження ефективності розробленого алгоритму.

Об’єкт дослідження – процес складання туристичних маршрутів.

Предмет дослідження – задача командного орієнтування з врахуванням часових вікон.

Методи дослідження, застосовані в роботі, базуються на методах дослідження операцій, паралельного програмування, метаевристичних та евристичних алгоритмах.

1.7 Висновки до розділу

Було наведено огляд існуючих задач у термінах яких можна сформулювати задачу складання туристичних маршрутів, в результаті чого, було обрано задачу командного спортивного орієнтування з врахуванням часових вікон. Для неї було проведено огляд існуючих методів розв‘язання і виявлено, що тема використання паралельних обчислень слабко досліджена.

З метою пошуку ефективного розв‘язання задачу командного спортивного орієнтування з врахуванням часових вікон, була сформульована відповідна постановка задачі та мета дослідження.

Слід зазначити, що дослідження направлені на розв‘язання задачі командного спортивного орієнтування з врахуванням часових вікон мають практичне значення не лише для проблеми складання туристичних маршрутів, а й для ряду задач, що пов‘язані з побудовою маршрутів для надання різноманітних послуг клієнтам. Було наведено короткий огляд цих задач. Левова частка програмного забезпечення, що створюється для автоматизації даних процесів, є мобільними застосуваннями, тому розробка ефективних і швидких алгоритмів для розв‘язання задач класу ОР є актуальною темою наразі.

2 РОЗВЯ'ЗУВАННЯ ЗАДАЧІ TOPTW

2.1 Змістовна постановка задачі

Необхідно побудувати набір туристичних маршрутів, щоб відвідати місця певної місцевості. Туристи перебувають у даній місцевості декілька днів. Час, що виділяється на відвідування місць ними є обмеженим кожен день. В результаті може виникнути ситуація, що вони не можуть відвідати всі місця за час їх візиту. Тому для місць задається оцінка, що позначає задоволеність туристів від їх відвідування. Таким чином, можна розставити пріоритети і формувати набір маршрутів з огляду них.

При побудові набору туристичні маршрутів вважається, що всі вони починаються і закінчуються в одному місці. Кінцевий і початковий пункти призначення не обов'язково повинні співпадати.

Необхідно при формуванні маршрутів враховувати ряд обмежень, що викликані часовими рамками. Не всі місця можна відвідати цілодобово. На переміщення між місцями і їх відвідування повинно відводитись достатньо часу. Також, може виникнути ситуація, коли треба витратити час на очікування відкриття певного місця. Для відвідування кінцевого і початкового пунктів час не відводиться.

Підсумовуючи, основною метою є побудова маршрутів на кожен день, що максимізується загальна сумарну задоволеність туристів від повного набору і враховуються всі вищенаведені обмеження.

В частині графічного матеріалу змістовну постановку задачі представлено на прикладі.

2.2 Математична постановка задачі

Математичним формулюванням наведеної змістовної постановки є задача Team Orienteering Problem with Time Windows (TOPTW). TOPTW – задача максимізації сумарної корисності набору маршрутів, що обмежені у часі.

Пункти (місця), що треба відвідати, можна представити як вершини орієнтовного повного зваженого графу. Граф містить $n, n \in N$ вершин. Необхідно

побудувати $m, m \in N$ маршрутів у цьому графі. Кожен маршрут починається з вершини $i = 1$, а закінчується в вершині $i = n$.

Математична постановка і її опис мають вигляд:

$$\sum_{d=1}^{m}\sum_{i=2}^{n-1} Score_i y_{id} \to max, \quad (2.1)$$

$$\sum_{j=2}^{n-1}\sum_{d=1}^{m} x_{1jd} = \sum_{i=2}^{n-1}\sum_{d=1}^{m} x_{ind} = m, \quad (2.2)$$

$$\sum_{i=1}^{n-1} x_{ikd} = \sum_{j=2}^{n} x_{kjd} = y_{kd}, k = \overline{2, n-1}, d = \overline{1, m}, \quad (2.3)$$

$$s_{id} + T_i + c_{ij} - s_{jd} \le M(1 - x_{ijd}),\ i = \overline{1, n},\ j = \overline{1, n}, d = \overline{1, m}, \quad (2.4)$$

$$\sum_{d=1}^{m} y_{kd} \le 1, \qquad k = \overline{2, n-1}, \quad (2.5)$$

$$\sum_{i=1}^{n-1}(T_i y_{id} + \sum_{j=2}^{n} c_{ij} x_{ijd}) \le T_{max},\ d = \overline{1, m}, \quad (2.6)$$

$$O_i \le s_{id}, \qquad i = \overline{1, n}, d = \overline{1, m}, \quad (2.7)$$

$$s_{id} \le C_i, \qquad i = \overline{1, n}, d = \overline{1, m}, \quad (2.8)$$

$$x_{ijd}, y_{id} \in \{0,1\}, i = \overline{1, n}, j = \overline{1, n}, d = \overline{1, m}, \quad (2.9)$$

$$C_n = T_{max}, \quad (2.10)$$

$$T_1 = T_n = 0. \quad (2.11)$$

Вхідні дані:

n – кількість пунктів;

m – кількість маршрутів, що треба побудувати;

$Score_i$ – корисність від відвідування пункту $i = \overline{1, n}$;

O_i – початок часового вікна, коли можна відвідати пункт $i = \overline{1, n}$;

C_i – кінець часового вікна, коли можна відвідувати пункт $i = \overline{1, n}$;

T_i – час необхідний на відвідування пункту $i = \overline{1, n}$;

T_{max} – максимальна тривалість кожного з маршрутів;

M – константа, значення якої велике відносно задіяних даних;

c_{ij} – час, що необхідний на переміщення між пунктами $i = \overline{1,n}$ та $j = \overline{1,n}$.

Змінними є:

s_{id} – час початку візиту до пункт $i = \overline{1,n}$ на маршруті $d = \overline{1,m}$;

x_{ijd} – змінна, що приймає значення 1, тільки коли на маршруті $d = \overline{1,m}$ пункт $j = \overline{1,n}$ відвідується одразу за пунктом $i = \overline{1,n}$.

Вихідні дані:

$$y_{id} = \begin{cases} 1, \text{якщо на маршруті } d \in \overline{1,m} \text{ відвідується пункт } i \in \overline{1,n} \\ 0, \text{ в іншому випадку} \end{cases}$$

Цільова функція:

(2.1) – сумарна корисність від відвідування всіх пунктів, що включаються в маршрути максимізується.

Обмеження:

(2.2) – початковий і кінцевий пункти повинні бути відвідані рівно m разів;

(2.3) – правило збереження потоку;

(2.4) – забезпечує, що проміжок часу між двома послідовними пунктами достатньо довгий, щоб виділити час на відвідування першого з них і переміщення між першим і другим пунктами;

(2.5) – забезпечує, що пункт не буді відвідано більше разу на всіх маршрутах;

(2.6) – забезпечує, що загальна тривалість кожного маршруту не більше T_{max};

(2.7, 2.8) – відвідування пункту повинно початись в рамках заданого часового вікна;

(2.10) – кінцевий пункт можна відвідати не пізніше T_{max};

(2.11) – кінцевий і початковий пункти не потребують часу для відвідування.

Розв'язком задачі є:

- розбиття множини вернин графу на підмножини (маршрути);
- задання порядку обходу на кожній підмножині.

Розв'язок є прийнятним (допустимим), якщо всі маршрути задовольняють обмеженням задачі.

2.3 Алгоритм детермінованого локального пошуку

Алгоритми детермінованого локального пошуку (ДЛП) - це сім'я ітераційних методів, заснована на частковому перебиранні варі антів на кожній ітерації серед точок околу поточної точки, тобто серед сусідніх до неї. В алгоритмах цього типу замість повного перебору застосо вується спрямований локальний перебір у підмножинах варіан тів, які називаються око лами. Цим пояснюється їх назва - алгоритми локального пошуку (Local search). У сфері комбінаторної оптимізації (КО) алго ритми ЛП мають давню історію через свою наочність і високу ефективність. Наприклад, перший алгоритм локального пошуку для задачі комівояжера був запропонований ще в 1956 р., а ло кальний пошук для задачі розміщення обладнання був розроб лений у 1962 р. Загальна схема ЛП у задачах мінімізації така: починаючи з деякого припустимого розв'язку задачі, новий роз в'язок із кращим значенням цільової функції шукають у його околі. Якщо такий розв'язок знайдено, то він приймається і по шук поліпшення розв'язку далі здійснюється вже в його околі і т. д. Алгоритм закінчується, коли досягнуто локального опти муму, тобто коли в околі поточного розв'язку немає ніякого ін шого варіанта з меншим значенням цільової функції. Загальна схема алгоритмів детермінованого локального по шуку може бути подана таким чином:

КРОК 1. Генерація початкового припустимого розв'язку x, який обираємо як поточний варіант.

КРОК 2. Чергова ітерація.

Формуємо окіл $O(x)$ поточного варіанта й точно чи наближе но знаходимо елемент $y \in O(x)$, який є субоптимальним розв'яз ком у цьому околі. Якщо $y \neq x$, то знайдений елемент оголошується черговим поточним варіантом (здійснюється переприсвоєння $x \leftarrow y$) і починається чергова ітерація, інакше - повернення на п. 3.

КРОК 3. Завершення роботи алгоритму: x - локальний розв'язок, якщо на останній ітерації здійснюється вичерпний пошук в околі [81].

Отже, у результаті роботи алгоритму локального пошуку при заданому початковому наближенні x^0 утворюється послідовність точок $x^1, \dots, x^m$, яка має такі характеристики:

– розмірність m послідовності є невідомою заздалегідь;

– кожен наступний розв’язок в послідовності належить околу попереднього;

– значення цільової функції кожного наступного розв’язку краще за попереднє;

– x^m є локальним розв’язком із зони притягання x^0.

Під зоною притягання розуміємо таку частину простору допустимих розв’язків, у якій цільова функціє є унімодальною. Локальний пошук подібний простому алгорит муспуску з тією відмінністю, що

– околи поточного розв'язку досліджуються систематично замість безладного блукання;

– пошук в околі повторюється до тих пір, поки не буде знайдений локально оптимальний розв'язок.

2.3.1 Ключові аспекти реалізації детермінованого локального пошуку

Принциповими моментами реалізації конкретних алгоритмів локального пошуку є:

– визначення околів $O(x)$;
– генерація чергової точки $y \in O(x)$;
– критерій завершення перегляду точок у поточному околі та переходу до наступного;
– спосіб обчислення величини зміни цільової функції при переході до нового поточного варіанта;
– критерій завершення;
– формування початкового наближення.

Ефективність алгоритмів локального пошуку істотно залежить від вибору відповідного типу околу $O(x)$. Чим більше окіл, тим імовірніше отримати кращий результат, але розширення околів швидко стає непрактичним. Утім для багатьох задач КО алгоритми з околами розміром більше, ніж $O(n^2)$, де n позначає розмі-

рність задачі, стають неефективними й тому досить рідко використовуються на практиці.

Найчастіше за все використовуються метричні околи чи околи, утворені алгоритмічно (як, наприклад, у алгоритмі 2-замін Ліна).

Можливі такі стратегії переходу в околі:

– до першого поліпшення за значенням цільової функції;

– до найкращого (перебирається весь окіл і відбирається найкраща точка).

Отже, пошук в околі може здійснюватися або шляхом повного перебору сусідніх точок (і тоді знайдене поліпшення - це найкра щий в околі варіант), або ж перехід здійснюється до першого знай деного варіанта, який поліпшує цільову функцію. Накопичений багаторічний досвід застосування алгоритмів локального пошуку свідчить, що повний перегляд околу на кожній ітерації потребує значних затрат часу, хоча й не гарантує знаходження в підсумку точнішого розв'язку, тому ефективнішим є пошук в околі до першого поліпшення. У деяких модифікаціях методів локального по шуку використовуються околи зі змінюваним радіусом.

Іншим аспектом, який суттєво впливає на ефективність алгоритмів локального пошуку, є організація перебору точок в околах. Можливі два способи організації перебору точок в околі при переході від ітерації до ітерації:

– лінійний генератор - при зміні поточного варіанта в ново му околі починається породження елементів околу з початку;

– кільцевий генератор - точки в околі вважаються певним чином упорядкованими й після переходу до нової ітерації про довжується перебір точок, починаючи з наступної відповідно до зазначеного упорядкування.

В обох випадках явно чи програмно визначається упорядкування операторів зсуву, які породжують точки в околі, і послідовно формуються такі точки. Однак у першому випадку після переходу до кращого варіанта перебір у новому околі починається з початкового оператора зсуву, а в другому - продовжу ється з поточного оператора зсуву. Відповідно ці способи дістали назву лінійного та кільцевого генератора.

Перевагою алгоритмів локального пошуку над багатьма ін шими евристиками є те, що простір варіантів може бути дослі джений дуже ефективно: замість того, щоб обчислювати значення цільової функції для $y \in O(x)$, достатньо обчислити тільки різницю $\Delta = f(x) - f(y)$, використовуючи властивості прикладної задачі, щоб уникнути явного обчислення $f(x)$ та $f(y)$, і перевірити, чи є величина Δ додатною.

Іншими словами, алгоритм ЛП здатний опрацювати n розв'язків у просторі пошуку за той самий час, за який, наприклад, еволюційний алгоритм оцінює тільки окремий розв'язок. ЛП використовує цю перевагу - як і інші алгоритми пошуку в околах, включаючи описані нижче.

Критерієм завершення часто вважають такі умови:

– відсутність поліпшуючої точки в околі поточного розв'язку, тобто перегляд усіх точок околу не завершився поліпшенням;

– проведення наперед заданої кількості ітерацій;

– вичерпання заданого ліміту часу.

За початковий варіант розв'язку найчастіше беруть довільний припустимий розв'язок. Для його знаходження часто використовують випадкове генерування варіантів розв'язку з подальшою перевіркою їх на припустимість - інколи таких варіантів генерується кілька, а з них вибирається найкращий. Часто також використовуються розв’язки, знайдені простими евристичними алгоритмами; при цьому намагаються максимально врахувати специфіку задачі.

Зазначимо, що обмеження задачі логічно враховувати при знаходженні чергового варіанта розв'язку, перевіряючи його на припустимість.

У комбінаторних просторах окіл часто визначається з вико ристанням метрик [81]. У деяких застосуваннях використовуються спеціальні метрики, які враховують особливості задач. Наприклад, для ЗК поняття сусідства може вводитися на основі 2-замін Ліна чи вставлення фрагментів.

До методів локального пошуку передусім належать:

– пошук (спуск) в околах (локальна оптимізація);

- пошук зі змінними околами;
- керований локальний пошук;
- табу-пошук.

2.4 Специфіка детермінованого локального пошуку для TOPTW

Далі під локальним пошуком (ЛП) будемо вважати детермнінований локальний пошук. Наведемо основні ознаки ЛП для TOPTW

Окіл поточного розв'язку формується з розв'язків, що утворюються шляхом додавання однієї вершини, що не включається в розв'язок. Окіл включає всі допустимі варіанти розміщення цієї вершини.

Початковим розв'язком є множина маршрутів, що складаються з початкової і ікінцевої вершин.

Чергова точка генерується шляхом включення однієї вершини в «найвигіднішу» позицію. До значення цільової функції додається корисність вставленої вершини. Після включення нової вершини новий розв'язок стає поточним і розглядається окіл нового розв'язку.

ЛП завершується, коли не можливо додати нову вершину в маршрути через часові обмеження на тривалість маршрутів та часових вікон.

Перехід в околі відбувається коли знаходиться «найвигідніша» позиція. Детальніше це описано при описі алгоритму вставки нової вершини. Перебір точок відбувається за принципом лінійного генератору.

2.5 Метод повторюваного локального пошуку

У якості методу розв'язання було обрано повторюваний локальний пошук (Iterated local search, ILS). Основною ідеєю, що лежить в основі ILS, є зосередження пошуку не на всьому просторі пошуку, а на розв'язках, що формуються деяким алгоритмом, що лежить в основі. Як правило, вбудованим алгоритмом є локальний пошук (Local search, LS). Отриману поведінку ILS можна охарактеризувати як ітеративну побудова ланцюга рішень вбудованого алгоритму. Результатом є концептуально проста метаевристика, яка, тим не менш, призвела до появи сучасних

ефективних алгоритмів для багатьох складних обчислювальних задач. Зазвичай, дуже хороша продуктивність часто досяжна навіть при досить простих реалізаціях метаевристики [44].

Крім того, модульна архітектура ILS робить його дуже підходящим для інженерного підходу у побудові алгоритму, коли продуктивність алгоритму поступово може бути додатково оптимізована.

Проста ідея, що лежить в основі ILS мала багато інтерпретацій в літературі, тому можна зустріти різні назви, такі як Iterated descent [14, 15], Large-step Markov chains [48], Iterated Lin-Kernighan [36], Chained local optimization [47] і їх комбінації [10]. Детальний огляд історіїї ILS наведено у [37].

2.5.1 Загальна схема алгоритму ILS

Для розгляду загальної схеми ILS вважатимемо, що вбудованим алгоритмом є LS. Нехай f – цільова функція, задачею комбінаторної оптимізації є мінімізація цільової функції. Допустимий розв‘язок задачі – s, множина всіх допустимих розв‘язків – S. Нехай в результаті роботу LS для s генерується детермінованим чином розв‘язок s^*, причому, значення $f(s^*) \leq f(s)$. Розв‘язки, що генеруються процедурою LS формують множину S^*. На рисунку 2.1 наведено розподіл щільності ймовірностей для $f(s^*)$ та f(s) для комбінаторних задач. Дисперсія та математичне сподівання для $f(s^*)$ менші ніж для f(s). Тому проводити пошук множині S^* ефективніше, ніж в S. Елементи S^* не відомі, тому визначити окіл і застосовувати LS на множині нема можливості. Замість цього, використовується алгоритм ILS. Будується послідовність значень, що належать S^*. Спочатку послідовність має тільки одне значення – s^*, тож саме воно є поточним на першій ітерації. Щоб отримати наступне значення послідовності, маючи поточне значення, виконується збурення s^*, внаслідок цього формується розв‘язок s', що належить S. З нього генерується новий розв‘язок $s^{*\prime}$, що належить S^*. Якщо даний розв‘язок задовольняє певному критерію прийнятності, то він стає поточним значенням послідовності і буде використаний для пошуку наступного значення послідовності. У протилежному випадку послідовність не змінюється, процедура знову повторюється

для s^*. Важливо, щоб збурення, що вносяться в поточний елемент послідовності були вагомими. Якщо зміни не вагомі, то ILS може згенерувати той самий поточний елемент, до якого вносились зміни. Занадто радикальні зміни можуть призводили до того, що ILS матиме природу випадкового пошуку. Тобто, коли випадковим чином генеруються значення з S для кожного з них виконується локальний пошук і обирається найкращий розв‘язок.

Оскільки детерміновані збурення можуть призводити до циклів, то необхідно робити їх адаптовними до зациклення або рандомізованими. Якщо перестановки залежать від попередніх значень з S^*, то формується послідовність у S^* з пам‘яттю .

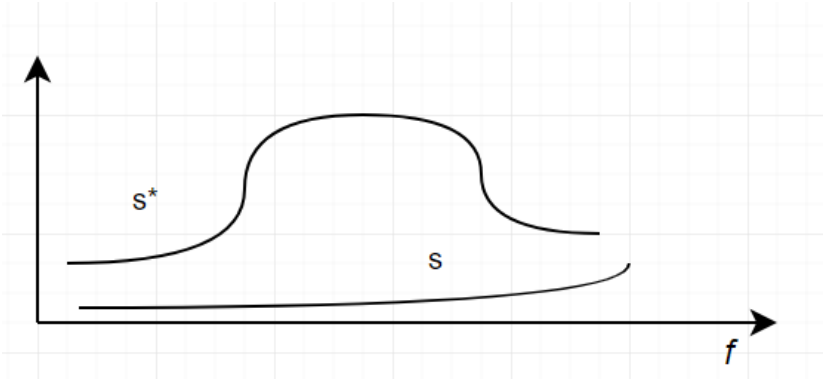


Рисунок 2.1 – Щільність розподілу значень цільової функції

Вищезгаданий критерій прийнятності визначає природу і ефективність побудованої послідовності розв‘язків, що належать S^*. Даний критерій регулює баланс між інтенсифікацією та диверсифікацією пошуку. Наприклад при ILS без пам‘яті для інтенсифікації пошуку застосовний критерій, що приймає новий розв‘язок лише коли він призводить до покращення значення цільової функції. Для інтенсифікації пошуку використовується критерій, що завжди приймає розв‘язок. Існують інші критерії, що ставлять на меті встановлення балансу між інтенсифікацією та диверсифікацією. Наприклад, розв‘язки, що призводять до покращення значення цільової функції приймаються завжди, а всі інші –з деякою ймовірністю. Інший можливий критерій з застосуванням пам‘яті – критерій з перезапуском. У випадку, коли певна кількість ітерацій ILS не призводить до покращення, то пошук починається заново з нового початкового розв‘язку. Вся послідовність розв‘язків в S^* конструюється заново.

Псевдокод алгоритму наведено на рисунку 2.2, де змінними є:

s^0 – початковий розв‘язок;

s^* - розв‘язок, що генерується вбудованим алгоритмом і додається до послідовності розв‘язків в S^*;

s' - розв‘язок, що отримується з s^* після збурення;

$s^{*\prime}$ - розв‘язок, що генерується вбудованим алгоритмом з s';

history – змінні, що служать за пам‘ять алгоритму.

Функціями є:

GenerateInitialSolution – генерує початковий розв‘язок;

LocalSearch – вбудований алгоритм, що проектує значення множини S в S^*;

Pertrubation – функція збурення;

AcceptanceCriterion – визначає чи буде додано розв‘язок до послідовності розв‘язків в S^*.

1: $s_0 = \text{GenerateInitialSolution}$
2: $s^* = \text{LocalSearch}(s_0)$
3: **repeat**
4: $\quad s' = \text{Perturbation}(s^*, history)$
5: $\quad s^{*\prime} = \text{LocalSearch}(s')$
6: $\quad s^* = \text{AcceptanceCriterion}(s^*, s^{*\prime}, history)$
7: **until** termination condition met

Рисунок 2.2 – Узагальнений алгоритм ILS

2.6 Специфіка методу повторюваного локального пошуку для TOPTW

Блок-схема алгоритму повторюваного локального пошуку наведена в частині графічного матеріалу.

Для розв’язування задачі TOPTW застосовується спеціальний вбудований алгоритм і алгоритм збурення розв’язку. Новий розв’язок приймається, якщо значення цільової функції краще за попереднє.

Крім змінних, що наведені у математичні постановці задачі, вводяться додаткові змінні:

$$a_i = s_j + T_j + c_{ji}, \; i, j = \overline{1, n}, \qquad (2.12)$$

$$\mathrm{Wait_i} = \max\{0, \mathrm{O_i} - \mathrm{a_i}\}, \quad \mathrm{i} = \overline{1,\mathrm{n}}, \tag{2.13}$$

$$\mathrm{MaxShift_i} = \min[\mathrm{MaxShift_{i+1}} + \mathrm{Wait_{i+1}}, \mathrm{C_i} - \mathrm{s_i}], \quad \mathrm{i} = \overline{1,\mathrm{n}}, \tag{2.14}$$

$$\mathrm{Shift_j} = \mathrm{c_{ij}} + \mathrm{c_{jk}} - \mathrm{c_{ik}} + \mathrm{Wait_j} + \mathrm{T_j}, \quad \mathrm{i,j,k} = \overline{1,\mathrm{n}}, \tag{2.15}$$

$$\mathrm{Ratio_i} = \frac{\mathrm{Score_i^2}}{\mathrm{Shift_i}}, \quad \mathrm{i} = \overline{1,\mathrm{n}}, \tag{2.16}$$

$$\mathrm{s_i} = \mathrm{a_i} + \mathrm{Wait_i}. \tag{2.17}$$

Змінна $\mathrm{a_i}$ позначає час прибуття до пункту (вершини) $\mathrm{i} \in \overline{1,\mathrm{n}}$. Її значення визначається за формулою (2.12), передбачається, що пункт j, $\mathrm{j} = \overline{1,\mathrm{n}}$, розташований безпосередньо перед пунктом i.

Оскільки прибуття в пункт може відбутись перед відкриттям, вводиться змінна $\mathrm{Wait_i}$, що позначає час очікування у пункті. Значення, що приймає змінна визначається за формулою (2.13).

Змінна $\mathrm{MaxShift_i}$позначає те, наскільки візит до пункту може буте відкладений, щоб такий зсув не привів до того, що час затрачений на весь маршрут не перевищував $\mathrm{T_{max}}$. Згідно формули (2.14), значення змінної дорівнює часу, тому наскільки візит до пункту, що розташований за даним у маршруті, може бути відкладений і часу, що треба очікувати перед початком наступного маршруту. В формулі також враховується те, що початок візиту завжди повинен знаходитись у рамках часового вікна, тому величина $\mathrm{MaxShift_i}$ не повинна перевищувати $\mathrm{C_i} - \mathrm{s_i}$.

Опис змінної $\mathrm{Ratio_i}$, що визначається за формулою (2.16), буде наведено нижче.

2.6.1 Побудова початкового розв’язку

Побудова початкового розв’язку починається з того, що всі маршрути містять тільки початкову і кінцеві вершини. Потім крок за кроком, додаються по одній новій вершині.

Для вибору нової вершини j, що буде потенційно додана у маршрут, обчислюється мінімальне значення змінної $\mathrm{Shift_j}$. Перевіряється кожна позиція, де може бути додана нова вершина у всіх маршрутах. По формулі (2.15) обчислюється

час, що буде затрачено, якщо між вершиною i та k буде додано нову вершину j. Щоб забезпечити допустимість отримуваних розв'язків, положення для яких значення $Shift_j$ більше за $MaxShift_i$ вважаються недопустимими. Таким чином, серед всіх допустимих положень обирається одне, де значення $Shift_j$ найменше. Після цього, за формулою (2.16) розраховується $Ratio_i$. В результаті обирається вершина у якої $Ratio_i$ найменше, вона буде додана до маршруту у положення, з найменшим зсувом маршруту у часі. Такі вставки у маршрут продовжуються поки можна знайти хоч одну вершину для якої є позиція, така що $Shift_j \leq MaxShift_j$.

2.6.2 Алгоритм вставки нової вершини

КРОК 1. Для кожної вершини j, що не належить жодному маршруту:

1.1 визначити позицію з найменшим $Shift_j$;

1.2 розрахувати $Ratio_j$.

КРОК 2. Вставити вершину з найбільшим $Ratio_j$ в маршрут.

КРОК 3. Розрахувати $Wait_j$, a_j, s_j для вставленої вершини.

КРОК 4. Для кожної вершини i , що слідує за вставленою:

4.1 Поки $Shift_i > 0$

4.2 оновити величини $Wait_i$, a_i, $MaxShift_i$, s_i.

КРОК 5. Оновити $MaxShift_j$ вставленої вершини.

КРОК 6. Для всіх вершин i перед вставленою:

6.1 оновити $MaxShift_i$

Для оновлення значень після вставки використовуються формули (2.15, 2.18-2.22). В даних формулах застосовується така нотація: нова вершина j вставляється між вершинами i та k. Для розрахування того, наскільки візит в вершину j збільшить загальну тривалість маршруту використовується формула (2.15). Отримується величина $Shift_j$, за цим значенням новий час очікування $Wait_{k*}$ для вершини (пункту) k розраховується за формулою (2.18) і новий час прибуття a_{k*} розраховується за формулою (2.19). $Shift_k$, що розраховується за формулою (2.20)

використовується для розрахування нових значень часу візиту і $\mathrm{MaxShift}_{k*}$ в вершині k за формулами (2.21) і (2.22) відповідно. Формули (2.18-2.22) також використовуються для оновлення змінних для вершин після k.

$$\mathrm{Wait}_{k*} = \max\left[0, \mathrm{Wait}_k - \mathrm{Shift}_j\right], \quad (2.18)$$

$$a_{k*} = a_k + \mathrm{Shift}_j, \quad (2.19)$$

$$\mathrm{Shift}_k = \max\left[0, \mathrm{Shift}_j - \mathrm{Wait}_k\right], \quad (2.20)$$

$$s_{k*} = s_k + \mathrm{Shift}_k, \quad (2.21)$$

$$\mathrm{MaxShift}_{k*} = \mathrm{MaxShift}_k - \mathrm{Shift}_k. \quad (2.22)$$

Описаний алгоритм застосовується не лише для побудови початкового розв'язку, він також використовується на кожній ітерації повторюваного локального пошуку для конструювання нового розв'язку.

2.6.3 Збурення розв'язку

Коли алгоритм вставки вершини завершує свою роботу, то ймовірно, що отриманий розв'язок є локальним оптимумом, тому необхідно провести модифікацію отриманого розв'язку, щоб мати можливість покращити знайдений розв'язок. Модифікація розв'язку відбувається шляхом вилучення $R \in N$ вершин на позиції $S \in N$ у кожному маршруті. При кожній ітерації повторюваного локального пошуку позиція S зсувається вперед. Коли S досягає кінця найкоротшого маршруту, то S поміщується у початок всіх маршрутів.

Після видалення вершини, всі візити для наступних вершин зсуваються у часі до вершини на позиції $S - 1$. Для вершини після видалених треба оновити Wait, a, s, MaxShift . Для вершин, що передують видалим необхідно оновити MaxShift.

Алгоритм збурення розв'язку

КРОК 1. Для кожного маршруту:

1.1 видалити R вершин починаючи з позиції S;

1.2 розрахувати Shift.

КРОК 2. Для кожної вершини і , що слідує за видаленою(ими):

2.1 Поки $\mathrm{Shift}_i > 0$

2.2 оновити величини $\mathrm{Wait_i}$, $\mathrm{a_i}$, $\mathrm{MaxShift_i}$, $\mathrm{s_i}$.

КРОК 3. Для всіх вершин і перед видаленими:

3.1 оновити $\mathrm{MaxShift_i}$

2.7 Модифікація алгоритму ILS

Модифікація оригінального алгоритму [35] полягає у тому, що змінюється критерій прийняття нового розв'язку. Замість того, щоб приймати всі розв'язки приймаються тільки не гірші.

Збереження попереднього розв'язку достатньо ресурсозатратне, тому що необхідно зберігати значення всіх змінних, що обчислені для вершин у маршрутах. Замість цього зберігаються вершини, що видалені на кроці збурення і ті, що додані на кроці вставки. Якщо розв'язок гірший, то тоді необхідно реконструювати старі маршрути. Спочатку видаляються вершини, що були вставлені при роботі ЛП (крок вставки). Перераховуються змінні, додаються видалені вершини і знову перераховуються значення змінних.

Блок-схема модифікованого алгоритму повторюваного локального пошуку наведена в частині графічного матеріалу.

2.8 Паралельне обчислення околу

На кроці вставки нової вершини, у розв'язок послідовно для кожної вершини, що не входить до маршрутів, визначається найкраща позиція для вставки. Цю процедуру можна виконувати паралельно на загальній пам'яті, адже при ній не може виникнути умови змагання (race condition). Кожній вершині можна виділити окремий процес. Йому треба модифікувати тільки «свої» ділянки пам'яті. Якщо він звертається до інших комірок пам'яті, то тільки для зчитування.

2.9 Приклад розв'язання задачі

Наведемо приклад побудови початкового розв'язку для конкретної задачі. Необхідно побудувати 2 маршрути з $\mathrm{T_{max}} = 170$. Дано 5 пунктів. З них 1 являє

собою пункт в якому закінчуються і починаються всі маршрути. Даний пункт матиме номер 0, крім того додається дві копії даного пункту. Вони служитимуть кінцевим пунктом для маршрутів і матимуть номери 5 та 6. Вхідні дані наведено у таблиці 2.1.

Таблиця 2.1 – Вхідні дані

Номер вершини i	Час на відвідування вершини T_i	Корисність від відвудування $Score_i$	Початок часового вікна, O_i	Кінець часового вікна C_i	Координата X	Координата Y
0	0	0	0	170	0	0
1	5	30	7	145	65	34
2	12	30	47	130	1	-46
3	10	10	0	180	-76	45
4	4	1	0	160	12	22
5	0	0	0	170	0	0
6	0	0	0	170	0	0

Час на переміщення між пунктами не задано, в даній задачі він прирівнюється до евклідових відстаней між вершинами. Розраховані значення наведено в таблиці 2.2.

Таблиця 2.2 – Час на переміщення між вершинами

№ вершини	0	1	2	3	4	5
0	0	73.36	46.01	88.32	25.05	0
1	73.35	0	102.45	141.42	54.34	73.35
2	46.01	102.4	0	119.2	68.88	46.01
3	88.32	141.4	119.2	0	90.95	88.32
4	25.05	54.34	68.88	90.95	0	25.05
5	0	73.36	46.01	88.32	25.05	0
6	0	73.36	46.01	88.32	25.05	0

Конструювання маршрутів починається з того, що в маршрути додаються початкова і кінцева вершини. Для кожної вершини розраховується змінні $\mathrm{Wait_i}$, $\mathrm{a_i}$, $\mathrm{MaxShift_i}$, $\mathrm{s_i}$. На даному етапі розв'язок має такий вигляд:

Маршрут 1: $0 \rightarrow 5$

Маршрут 2: $0 \rightarrow 6$

Значення змінних для вершини 0:

- $\mathrm{a} = 0;$
- $\mathrm{s} = 0;$
- $\mathrm{Wait} = 0;$
- $\mathrm{MaxShift} = 0.$

Значення змінних для вершин 5 і 6:

- $\mathrm{a} = 0;$
- $\mathrm{s} = 0;$
- $\mathrm{Wait} = 0;$
- $\mathrm{MaxShift} = 170.$

Крок 1

Необхідно визначити вершину, яка буде вставлена в один з маршрутів наступною. Вона обирається з тих вершин, що поки не належать жодному з маршрутів: 1, 2, 3, 4. Вибір робиться на основі величини Shift, що розраховується по формулам (2.14, 2.15). Результати розрахунків наведено у таблиці 2.3. Номер позиці – потенційне місце вставки нової вершини. Наприклад, якщо в таблиці № позиції дорівнює 1, то це означає, що вставка відбудеться після першої по порядку вершини у маршруті. Перша вершина у маршруті має номер 0. Пусті місці у таблиці означають, що дану вершину неможливо вставити в вказану позицію маршруту. Така ситуація може виникнути з двох причин: прибуття у пункт відбувається після закриття; вставка неможлива, бо час необхідний для цього більше ніж MaxShift, іншими словами, зсув приведе до того, що розклад стане недопустимим. Розклад недопустимий, якщо пункти відвідується не в рамках своїх часових вікон, чи загальна тривалість маршруту перевищує $\mathrm{T_{max}}$.

Таблиця 2.3 – Розрахунки для визначення найкоротшого часу вставки для кожної вершини на кроці 1

№ вершини, що вставляється	№ маршруту	№ позиції	Shift
1	1	1	151
1	2	1	151
2	1	1	105
2	2	1	105
3	1	1	-
3	2	1	-
4	1	1	54
4	2	1	54

Після даних обчислень для кожної вершини запам’ятовується найменше значення змінної Shift і відповідні значення позиції і маршруту вставки. Збережені дані відображено в таблиці 2.4. На наступному кроці серед даних вершин буде обрана одна, що має найбільше значення Ratio. Дані значення обчислюються згідно формули 2.16. Результати обчислень наведено в таблиці 2.4.

Таблиця 2.4 – Результати обчислень Ratio на кроці 1

№ вершини, що вставляється	№ маршруту	№ позиції	Мінімальне значення Shift	Значення Ratio
1	1	1	151	5.96
2	1	1	105	8.57
3	-	-	-	-
4	1	1	54	0.018

З таблиці 2.4 видно, що найбільше значення Ratio вершини 2. Тому, вершина 2 буде вставлена у перший маршрут і стане другою вершиною по порядку в ньому. Після вставки для вершини 2 обчислюються значення змінних Wait, a, s за формулами (2.12, 2.13, 2.17). Далі за формулами (2.18-2.22) для вершини 5 оновлюються значення змінних Wait, a, s, MaxShift. Тепер можна проводити обчислення MaxShift для вершини 2.14.

В результаті розв’язок має такий вигляд:

Маршрут 1: $0 \rightarrow 2 \rightarrow 5$

Маршрут 2: $0 \rightarrow 6$

Значення змінних для вершин 0 та 6 не змінюються.

Значення змінних для вершини 2:

- $a = 46$;
- $s = 47$;
- $Wait = 1$;
- $MaxShift = 66$.

Значення змінних для вершини 5:

- $a = 105$;
- $s = 105$;
- $Wait = 0$;
- $MaxShift = 65$.

Тепер вершину 2 не можна вставити повторно в маршрут. Оскільки на даному етапі було додано вершину у один з маршрутів, пошук продовжується. Коли на поточному кроці не додано жодної вершини, то пошук завершується.

Крок 2

Необхідно визначити вершину, яка буде вставлена в один з маршрутів наступною. Вибір робиться поміж вершин 1, 3, 4 на основі величини Shift. Результати розрахунків наведено у таблиці 2.5.

Таблиця 2.5 - Розрахунки для визначення найкоротшого часу вставки для кожної вершини на кроці 2

№ вершини, що вставляється	№ маршруту	№ позиції	Shift
1	1	1	-
1	1	2	-
1	2	1	151
3	1	1	-
3	1	2	-

Продовження таблиці 2.5

№ вершини, що вставляється	№ маршруту	№ позиції	Shift
3	2	1	-
4	1	1	52
4	1	2	52
4	2	1	54

В таблиці 2.6 наведено розрахунки, на основі яких обиратиметься вершина для вставки.

Таблиця 2.6 – Результати обчислень Ratio на кроці 2

№ вершини, що вставляється	№ маршруту	№ позиції	Мінімальне значення Shift	Значення Ratio
1	2	1	151	5.9
3	-	-	-	-
4	1	1	52	0.02

Згідно розрахунків у таблиці 2.6 видно, що найбільше значення Ratio вершини 1. Тому, вершина 1 буде вставлена у другий маршрут і стане другою вершиною по порядку в ньому. Значення змінних обчислюються за таким же принципом, як на кроці 1.

В результаті розв’язок має такий вигляд:

Маршрут 1: $0 \rightarrow 2 \rightarrow 5$

Маршрут 2: $0 \rightarrow 1 \rightarrow 6$

Значення змінних для вершин 0, 2, 5 не змінюються.

Значення змінних для вершини 1:

- $\mathrm{a} = 73;$
- $\mathrm{s} = 73;$
- $\mathrm{Wait} = 0;$
- $\mathrm{MaxShift} = 19.$

Значення змінних для вершини 6:

- $a = 151;$
- $s = 151;$
- $\text{Wait} = 0;$
- $\text{MaxShift} = 19.$

Тепер вершину 1 не можна вставити повторно в маршрут. На даному етапі було додано вершину у один з маршрутів, тому пошук продовжується.

Крок 3

Вершина 3 або 4 потенційно може бути вставлена на даному кроці у один з маршрутів, вибір родиться на основі розрахунків, що наведені в таблицях 2.7, 2.8.

Таблиця 2.7 - Розрахунки для визначення найкоротшого часу вставки для кожної вершини на кроці 3

№ вершини, що вставляється	№ маршруту	№ позиції	Shift
3	1	1	-
3	1	2	-
3	2	1	-
3	2	2	-
4	1	1	52
4	1	2	52
4	2	1	10
4	2	2	10

Таблиця 2.8 – Результати обчислень Ratio на кроці 3

№ вершини, що вставляється	№ маршруту	№ позиції	Мінімальне значення Shift	Значення Ratio
3	-	-	-	-
4	2	1	10	0.1

Згідно розрахунків у таблиці 2.8 видно, що можливо вставити лише вершину 4. Вона буде вставлена у другий маршрут і стане другою вершиною по порядку в ньому. Значення змінних обчислюються за таким же принципом, як на кроці 1.

В результаті розв’язок має такий вигляд:

Маршрут 1: $0 \rightarrow 2 \rightarrow 5$

Маршрут 2: $0 \rightarrow 4 \rightarrow 1 \rightarrow 6$

Значення змінних для вершин 0, 2, 5 не змінюються.

Значення змінних для вершини 4:

- $\mathrm{a} = 25;$
- $\mathrm{s} = 25;$
- $\mathrm{Wait} = 0;$
- $\mathrm{MaxShift} = 9.$

Значення змінних для вершини 1:

- $\mathrm{a} = 83;$
- $\mathrm{s} = 83;$
- $\mathrm{Wait} = 0;$
- $\mathrm{MaxShift} = 9.$

Значення змінних для вершини 6:

- $\mathrm{a} = 161;$
- $\mathrm{s} = 161;$
- $\mathrm{Wait} = 0;$
- $\mathrm{MaxShift} = 9.$

На даному етапі було додано вершину у один з маршрутів, тому пошук продовжується.

Крок 4

На даному кроці можливо вставити лише вершину 3. Розрахунки найкоротшого часу вставки наведено у таблиці 2.9.

З таблиці видно, що неможливо вставити вершину 3 у будь-який маршрут, бо маршрут стане недопустимим після цього. Жодної вершини не вставлено, тому алгоритм завершує свою роботу. Початковий розв’язок побудовано.

Таблиця 2.9 - Розрахунки для визначення найкоротшого часу вставки для кожної вершини на кроці 4

№ вершини, що вставляється	№ маршруту	№ позиції	Shift
3	1	1	-
3	1	2	-
3	2	1	-
3	2	2	-
3	2	3	-

2.10 Алгоритм імітаційного відпалу

Алгоритм імітації відпалу (Simulated Annealing, SA) серед розроблюваних метаевристик був одним з перших методів, що надавали стратегію уникнення локального оптимуму. Метод розроблений на основі алгоритму Метрополіса з області статистичної механіки. Ідея SA була основана на процесі відпалу металу і скла, коли конфігурування низького рівня споживання енергії забезпечується при правильному графіку охолодження [8]. SA вперше був представлений як алгоритм пошуку для комбінаторних задач в [21, 39]. Збіжність методу доведено в [45].

Фундаментальна ідея уникнення потрапляння в локальні оптимуми полягає в тому, щоб дозволити рухатися до розв’язків, що мають гірші значення цільової функції ніж у поточного розв’язку s. На кожній ітерації формуються елементи околу $s` \in N(s)$. Якщо для конкретного розв’язку s' значення цільової функції краще, ніж для розв’язку s, то s' стає новим поточним розв’язком. В протилежному випадку s' приймається з імовірністю, що є функцією від початкової температури (параметр алгоритма) T_k і $f(s') - f(s)$. Зазвичай, дана імовірність розраховується за законом Больцмана-Ґібса:

$$p(s`, s, T) = e^{-\frac{f(s`)-f(s)}{T_k}}. \quad (2.23)$$

Динамічний процес SA породжує ланцюг Маркова, оскільки на кожній ітерації вибір нового розв'язку залежить лише від значення попереднього розв'язку. Це означає, що базова процедура SA не має пам'яті. Однак, використання пам'яті при дослідженні простору пошуку може бути застосовано для побудови ефективних

методів, що конструюються на основі SA. Приклад застосування SA з пам'яттю наведено у [22].

В структурі алгоритму можна виділити три основні частини:

– побудова початкового розв'язку;

– встановлення початкової температури;

– зміна температури.

В псевдокоді алгоритму, який наведено на рисунку 2.3, цим крокам у відповідність ставляться такі процедури:

– GenerateInitialSolution();

– SetInitialTemperature();

– AdaptTemperature(T_k).

```
s ← GenerateInitialSolution()
k ← 0
T_k ← SetInitialTemperature()
while termination conditions not met do
  s' ← PickNeighborAtRandom(N(s))
  if (f(s') < f(s)) then
    s ← s'          {s' replaces s}
  else
    Accept s' as new solution with probability p(s' | T_k, s)
  end if
  AdaptTemperature(T_k)
  k ← k + 1
end while
output: best solution found
```

Рисунок 2.3 – Псевдокод алгоритму імітації відпалу [22]

Алгоритм починається з створення початкового розв'язку, який може бути побудовано випадково або евристично. Початкова температура вибирається така, що ймовірність прийняття погіршуючих розв’язків була досить високою на початку роботи алгоритму. Температура T_k змінюється на кожній ітерації по температурі відповідно до графіку охолодження. Графік охолодження визначає значення T_k на кожній такій ітерації k. Вибір підходящого графіку охолодження має вирішальне значення для ефективної роботи алгоритму. Ймовірність прийняття погіршуючих

розв’язків повинна зменшуватись при роботі алгоритму, чим досягається немонотонний характер зміни цільової функції.

Теоретично, необхідно обирати графік охолодження таким чином, щоб імовірність знаходження глобального оптимуму при $k \to \infty$ збігалась до одиниці. Така збіжність забезпечується, коли графік охолодження слідує логарифмічному закону. На жаль, графіки охолодження, що забезпечують збіжність до глобального оптимуму, часто не застосовні практично, бо вони занадто повільні. Тому використовуються швидші графіки охолодження. Один з найбільш популярних графіків охолодження – графік, що слідує геометричному закону (2.24).

$$T_k = \alpha T_{k-1}, \quad \alpha \in (0,1). \tag{2.24}$$

Графік охолодження може використовуватися для балансування між диверсифікацією та інтенсифікацією пошуку. Наприклад, на початку пошуку, T_k може бути константним значенням або лінійно зменшується, щоб дослідити простір пошуку; після T_k може слідувати, наприклад, геометричному закону, щоб забезпечити швидшу збіжність алгоритму до локального оптимуму в кінці пошуку. Більш ефективним варіантом є застосування немонотонного графіку охолодження [45, 55]. Такий графік характеризується чергуванням фаз охолодження та нагрівання, що забезпечує коливальний баланс між диверсифікацією та інтенсифікацією.

Графік охолодження та початкова температура повинні бути адаптовані до конкретної проблеми, що розглядається, оскільки вартість уникнення локальних оптимумів залежить від ландшафту простору пошуку. Простий спосіб емпіричного визначення початкової температури - спочатку випадковим чином згенерувати розв’язки, щоб приблизно оцінити середнє значення та дисперсію значень цільової функції. Заснована на даних розв’язках, початкова температура визначається таким чином, що прийняття погіршуючих розв’язків має високу вірогідність.

У [39] запропоновано обирати початкову температуру як максимальне значення різниці між значеннями цільових функцій будь-яких двох сусідніх розв’язків (2.25).

$$T_0 = \Delta f_{max}. \quad (2.25)$$

Інша схема, що ґрунтується на більш точній оцінці розподілу запропонована у [7, 77]. Рекомендується обирати значення початкової температури згідно формули (2.26).

$$T_0 = K\sigma_\infty^2. \quad (2.26)$$

Тут σ_∞^2 - момент другого порядку розподілу значення цільової функції, коли температура наближається до ∞. σ_∞ оцінується на основі випадкової генерації розв'язків.

Більш класичний та інтуїтивно зрозумілий метод описаний у [39]. Він полягає у обчисленні такої температури, що коефіцієнт приймання погіршуючого розв'язку приблизно дорівнює заданому значенню χ_0. Спочатку обирається висока початкова температура. Тоді робиться декілька ітерацій з використанням цієї температури. Імовірність прийнятих переходів порівнюється з χ_0. Якщо вона менша за χ_0, температура збільшується в 2 рази. В протилежному випадку температура зменшується. Процедура триває, доки імовірність переходу не приближується до χ_0.

Інший підхід запропоновано в [16].Задається додатня початкова температура і на її основі робиться прогін SA для одного чи більше початкових розв'язків $s_n \in S, n \in N^+$. Потім відбираються погіршуючі розв'язки $s'_{nt} \in S', t \in N^+$. В результаті отримується множина пар розв'язків E, опис якої подається формулою (2.27):

$$E = \{(f(s), f(s')) | s \in S', s' \in S', f(s) > f(s'), f(s) \xrightarrow{SA} f(s')\}. \quad (2.27)$$

Проводиться розрахунок оцінки температури за формулою (2.7):

$$\widehat{\chi}(T_n) = \frac{\sum_{(f(s),f(s'))\in E} \exp\left(-\frac{f(s)}{T_n}\right)}{\sum_{f(s),f(s')\in E} \exp\left(-\frac{f(s')}{T_n}\right)}. \quad (2.28)$$

Процедура закінчується, якщо виконується умова (2.29), де χ_0-очікувана ймовірність прийняття погіршуючих розв'язків, ϵ- точність.

$$|\widehat{\chi}(T_n) - \chi_0| < \epsilon. \quad (2.29)$$

В протилежному випадку нова оцінка температури обчислюється за формулою (2.30), де p- константа і проводяться розрахунки для $\widehat{\chi}(T_{n+1})$ за формулою (2.28).

$$T_{n+1} = T_n\left(\frac{\ln(\widehat{\chi(T_n)})}{\ln(\chi_0)}\right)^p. \quad (2.30)$$

Збіжність даного алгоритму доведено в [16].

2.11 Специфіка алгоритму імітаційного відпалу задачі TOPTW

Початкова температура обирається на основі методу з [16], що описаний в кінці попереднього розділу. Процедура визначення початкової температури запускається з такими початковими параметрами:

- $\chi_0 = 0.8$;
- $P = 0.5$;
- $\epsilon = 0.1$;

Графік охолодження слідує логарифмічному закону з параметром $\alpha = 0.9$.

Процедури побудови і збурення розв'язку, такі ж як для алгоритму повторюваного локального пошуку. Змінюється критерій прийняття нового розв'язку. Новий розв'язок приймається з ймовірністю, що виначається за формулою (2.23).

2.12 Висновки до розділу

Було подано математичну і змістовну постановки задачі TOPTW та методи її розв’язання. Для обраних методів було наведено узагальнену схему алгоритму і описано специфіку реалізації даних методів для задачі TOPTW. Був наведений приклад розв’язання задачі TOPTW алгоритмом, що лежить в основі обраних методів. Було запропоновано модифікацію, що може лежати в основі обох алгоритмів.

3 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вхідні данні

Вхідні дані – це дані, які надходять до системи і які необхідні для її подальшої роботи. Вхідні данні зчитуються з файлів чи надаються користувачем. Приклад файлу з вхідними даними для роботи алгоритмів наведено на рисунку 3.1.

Щоб сформувати файл вхідних даних для задачі TOPTW користувач вводить такі дані:

– префікс на основі якого буде формуватись назва файлу з вхідними даними;

– інформацію про кожний пункт в топології (графі);

– час, що виділяється на весь маршрут.

Про кожен пункт топології необхідні такі дані:

– координати пункту;

– корисність від відвідування пункту;

– час, що необхідний для відвідування пункту;

– час відкриття і час завершення відвідування;

– кількість пунктів призначення;

– унікальний номер пункту.

Номери пунктів і їх кількість визначаються автоматично.

Для того, щоб запустити алгоритми необхідно надати такі вхідні дані:

– кількість маршрутів;

– обмеження на час роботи;

– назви алгоритму(ів);

– шлях до файлів з вхідними даними для роботи алгоритмів.

```
5 3 48 4
500 200
 0 -10.442 19.999 0 0 0 0 0 1000
 1 -29.730 64.136 2 12 4 1 15 354 509
 2 -30.664 5.463 7 8 4 1 15 234 401
 3 51.642 5.469 21 16 4 1 15 411 573
 4 -13.171 69.336 24 5 4 1 15 474 622
 5 -67.413 68.323 1 12 4 1 15 155 295
 6 48.907 6.274 17 5 4 1 15 361 509
 7 5.243 22.260 6 13 4 1 15 451 629
 8 -65.002 77.234 5 20 4 1 15 425 588
 9 -4.175 -1.569 7 13 4 1 15 72 199
 10 23.029 11.639 1 18 4 1 15 157 318
 11 25.482 6.287 4 7 4 1 15 296 444
 12 -42.615 -26.392 10 6 4 1 15 111 249
 13 -76.672 99.341 2 9 2 2 5 10 368 528
 14 -20.673 57.892 16 9 2 2 5 10 98 251
 15 -52.039 6.567 23 4 2 2 5 10 96 208
 16 -41.376 50.824 18 25 2 2 5 10 382 535
 17 -91.943 27.588 3 5 2 2 5 10 436 580
 18 -65.118 30.212 15 17 2 2 5 10 405 528
 19 18.597 96.716 13 3 2 2 5 10 255 414
 20 -40.942 83.209 10 16 2 2 5 10 293 470
 21 -37.756 -33.325 4 25 2 2 5 10 298 408
 22 23.767 29.083 23 21 2 2 5 10 479 607
 23 -43.030 20.453 20 14 2 2 5 10 376 536
 24 -35.297 -24.896 10 19 2 2 5 10 91 238
 25 -54.755 14.368 4 14 1 4 1 2 4 8 360 505
 26 -49.329 33.374 2 6 1 4 1 2 4 8 379 528
 27 57.404 23.822 23 16 1 4 1 2 4 8 258 428
 28 -22.754 55.408 6 9 1 4 1 2 4 8 352 509
 29 -56.622 73.340 8 20 1 4 1 2 4 8 288 380
 30 -38.562 -3.705 10 13 1 4 1 2 4 8 159 324
 31 -16.779 19.537 7 10 1 4 1 2 4 8 423 564
 32 -11.560 11.615 1 16 1 4 1 2 4 8 238 387
 33 -46.545 97.974 21 19 1 4 1 2 4 8 339 429
 34 16.229 9.320 6 22 1 4 1 2 4 8 397 572
 35 1.294 7.349 4 14 1 4 1 2 4 8 479 599
 36 -26.404 29.529 13 10 1 4 1 2 4 8 315 492
 37 4.352 14.685 9 11 1 4 1 2 4 8 132 290
 38 -50.665 -23.126 22 15 1 4 1 2 4 8 161 326
 39 -22.833 -9.814 22 13 1 4 1 2 4 8 387 508
 40 -71.100 -18.616 18 15 1 4 1 2 4 8 284 446
 41 -7.849 32.074 10 8 1 4 1 2 4 8 296 471
 42 11.877 -24.933 25 22 1 4 1 2 4 8 381 482
 43 -18.927 -23.730 23 24 1 4 1 2 4 8 401 522
 44 -11.920 11.755 4 3 1 4 1 2 4 8 432 564
 45 29.840 11.633 9 25 1 4 1 2 4 8 289 450
 46 12.268 -55.811 17 19 1 4 1 2 4 8 451 597
 47 -37.933 -21.613 10 21 1 4 1 2 4 8 123 302
 48 42.883 -2.966 17 10 1 4 1 2 4 8 98 233
```

Рисунок 3.1 – Приклад файлу вхідних даних

3.2 Вихідні данні

При роботі з ПЗ формуються такі вихідні дані:

– текстові файли з даними для роботи алгоритмів;

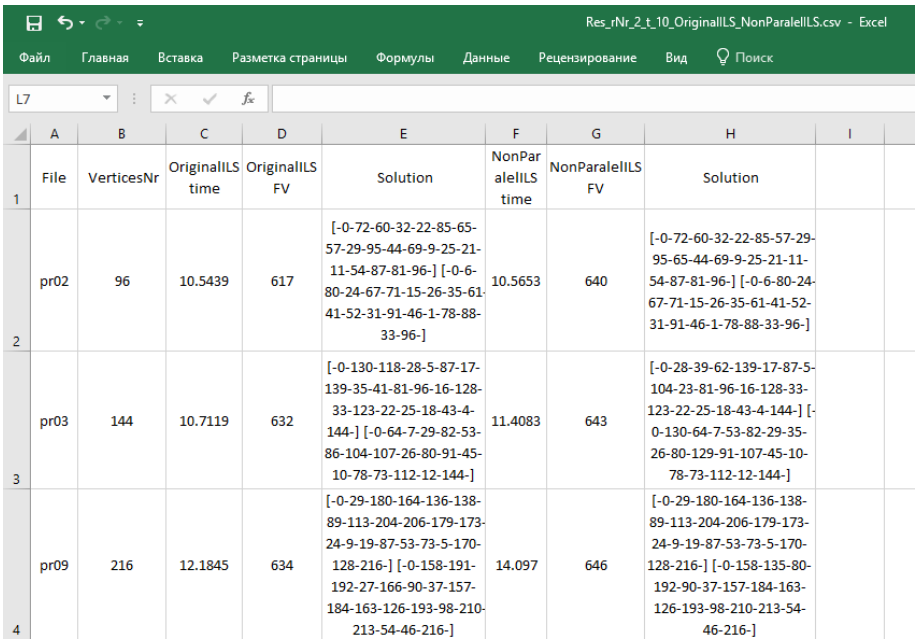
– Excel файли з результатами роботи алгоритмів.

Опис файлів з даними для роботи алгоритмів наведено у [53].

Результати роботи алгоритмів записуються в файл Excel. На рисунку 3.2 представлений приклад файлу вихідних даних, що сформовано ПЗ після роботи алгоритмів послідовного модифікованого повторюваного локального пошуку і немодифікованого повторюваного локального пошуку. В першій колонці наведено назву файлу вхідних даних на яких запусались алгоритми. В другій колонці

наведено кількість вершин. В наступних колонках для кожного алгоритму записано час роботи, обчислене значення цільової функції і сформовані маршрути.

Назва Excel файлу вихідних даних формується з таких даних: кількість маршрутів, обмеження на час роботи, назви алгоритмів.



	A	B	C	D	E	F	G	H	I
1	File	VerticesNr	OriginalILS time	OriginalILS FV	Solution	NonParalelILS time	NonParalelILS FV	Solution	
2	pr02	96	10.5439	617	[-0-72-60-32-22-85-65-57-29-95-44-69-9-25-21-11-54-87-81-96-] [-0-6-80-24-67-71-15-26-35-61-41-52-31-91-46-1-78-88-33-96-]	10.5653	640	[-0-72-60-32-22-85-57-29-95-65-44-69-9-25-21-11-54-87-81-96-] [-0-6-80-24-67-71-15-26-35-61-41-52-31-91-46-1-78-88-33-96-]	
3	pr03	144	10.7119	632	[-0-130-118-28-5-87-17-139-35-41-81-96-16-128-33-123-22-25-18-43-4-144-] [-0-64-7-29-82-53-86-104-107-26-80-91-45-10-78-73-112-12-144-]	11.4083	643	[-0-28-39-62-139-17-87-5-104-23-81-96-16-128-33-123-22-25-18-43-4-144-] [-0-130-64-7-53-82-29-35-26-80-129-91-107-45-10-78-73-112-12-144-]	
4	pr09	216	12.1845	634	[-0-29-180-164-136-138-89-113-204-206-179-173-24-9-19-87-53-73-5-170-128-216-] [-0-158-191-192-27-166-90-37-157-184-163-126-193-98-210-213-54-46-216-]	14.097	646	[-0-29-180-164-136-138-89-113-204-206-179-173-24-9-19-87-53-73-5-170-128-216-] [-0-158-135-80-192-90-37-157-184-163-126-193-98-210-213-54-46-216-]	

Рисунок 3.2 – Файл результатів роботи алгоритмів

3.3 Засоби розробки

Для написання програмного забезпечення використовувалось середовище розробки, що надається Visual Studio Community 2017.

Microsoft Visual Studio – серія продуктів, що розроблена Microsoft. Вони включають інтегроване середовище розробки програмного забезпечення разом з іншими інструментальними засобами для розробки ПЗ [73]. Дані продукти дозволяють розробляти як консольні додатки, додатки з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, веб-застосування, веб-сторінки, веб-служби як в некерованому [68], так і в керованому [75] кодах для всіх платформ, підтримуваних Windows, Windows CE, Windows Mobile, .NET Framework, Windows Phone .NET Compact Framework, Xbox і Silverlight.

Вбудований відладчик може застосовуватись для відладки вихідного коду і машинного коду. Редактори вихідного коду в Visual Studio підтримують технології

IntelliSense і надають інструменти для рефакторингу коду [17]. До інших вбудованих інструментів належать: дизайнер класів, дизайнер схем баз даних, редактор форм для спрощення створення графічних інтерфейсів, веб-редактор. При використанні Visual Studio можна підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні. Також за допомогою Visual Studio можна створювати власні плагіни.

Для розробки використовувалась остання безкоштовна версія Visual Studio - Visual Studio 2017 Community [49].

Як було зазначено вище, Visual Studio підтримує багато мов програмування і Windows-сумісних платформ. Для створення програмного забезпечення використовувалась платформа .NET Framework.

Однією з основних ідей Microsoft .NET є сумісність програмних частин, написаних на різних мовах. Наприклад, можна написати клас на C # і наслідувати його від класу на Visual Basic .NET [1, 76].

Разом з Microsoft Visual Studio поставляються(лись) такі мови: C #, Visual Basic .NET, JScript .NET, C ++ / CLI, F #, J #.

Завдяки тому, що фреймворк і Visual Studio надають можливості для інтергації програмних компонентів, що написані на різних мовах, програмне забезпечення було написано з використанням C ++ і C #.

C # – об'єктно-орієнтована мова програмування з безпечною системою типізації. Здебільшого C # використовується разом з фреймворком .NET [19]. C# розроблялась як мова програмування прикладного рівня для CLR [24] і тому вона залежить, перш за все, від можливостей самої CLR.

Засобами C# і .NET було створено графічний інтерфейс користувача. .NET надає простий інтерфейс програмування для цього - Windows Forms, що відповідає за графічний інтерфейс користувача. Даний інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows [52].

1. C # - це чисто об'єктно-орієнтована мова, в той час, як C++, наприклад, являє собою комбінацію об'єктно-орієнтованого та процедурно-орієнтованого

підходів. Це дозволяє створювати модульні легкопідтримувані програми, повторно використовувати код. Це одна з найбільших переваг C # над C ++.

2. Автоматичний збір сміття відсутній у C++. C # має ефективну систему для автоматичної очистки пам’яті. Це звільняє від необхідності контролю за можливими витоками пам’яті, коли в пам’яті зберігаються дані, що вже не використовуються, а програма не знає, де вони знаходяться.

3. Розробка легше і швидше, завдяки тому, що наявні бібліотеки роблять багато функцій легким для реалізації.

4. У C # наявні такі мовні функції, як лямбда-вирази, методи розширення, вирази-запити.

5. C # підтримує некерований код.

6. Програми, написані на C#/.NET високоінтегровні з іншими .NET-сумісними технологіями і мовами.

7. Формалізована концепція get-set сприяє тому, що код стає більш розбірливий. Крім того, в C # нема потреби турбуватись про включення заголовків.

8. Наявна велика кількість навчальних матеріалів і підтримку від Microsoft в C # (. NET Framework).

Для алгоритмічного забезпечення було створено Dynamic-link library (DLL) [27], що написана на мові C ++.

C ++ - компільована, статично типізована мова програмування загального призначення [20]. Парадигми програмування, що підтримуються мовою: процедурне програмування, об'єктно-орієнтоване програмування, узагальнене програмування.

Мова C++ добре підходить для створення математичного забезпечення ПЗ з ряду причин.

1. Висока сумісність з мовою C, що призводить до обчислювальної продуктивності. Мова спроектований таким чином, щоб дати програмісту максимальний контроль над усіма аспектами структури та порядку виконання програми. Присутня можливість роботи з пам'яттю на низькому рівні.

2. Підтримка різних стилів програмування: структурне, об'єктно-орієнтоване, узагальнене, функціональне програмування, породжуюче метапрограмування.

3. Автоматичний зворотній виклик конструкторів спрощує і підвищує надійність керування пам'яттю та іншими ресурсами.

4. Перевантаження операторів дозволяє коротко і легко записувати вирази над користувацькими типами в природній алгебраїчній формі.

5. Присутня можливість управління константністю об'єктів (модифікатори const, mutable);

6. Доступність вивчення - для C ++ існує величезна кількість навчальної літератури, яку переведено на всі можливі мови.

7. Наявність потужних технології для роботи з розподіленою і загальною пам‘яттю.

Як було зазначено, C++ надає можливість оптимізувати процес роботи програми з загальною пам’яттю завдяки підтримці стандарту OpenMP. Строрене ПЗ використовує директиви OpenMP.

OpenMP (Open Multi-Processing) - відкритий стандарт для розпаралелювання програм на мовах C, C ++ і Фортран [54]. Дає опис сукупності директив компілятора, бібліотечних процедур і змінних середовища виконання, які призначені для програмування багатопотокових додатків на багатопроцесорних системах із загальною пам'яттю. Передбачається, що потоки виконуються паралельно на машині з декількома процесорами (кількість процесорів не обов'язково має бути більше або дорівнює кількості потоків).

Завдання, що виконуються потоками паралельно, також як і дані, необхідні для виконання цих завдань, описуються за допомогою спеціальних директив препроцесора відповідної мови. Кількість створених потоків може регулюватися як самою програмою за допомогою виклику бібліотечних процедур, так і ззовні, за допомогою змінних оточення.

Наразі існує 5 версів OpenMP. Visual Studio надає підтримку для версії OpenMP 2.0, тому її було використано при написанні ПЗ.

3.4 Архітектура програмного забезпечення

Структура проекту ПЗ наведена на рисунку 3.3. Основними компонентами проекту є:

– компонент TOPTW, що являє собою динамічну бібліотеку;

– компонент TOPTWLib, що являє собою статичну бібліотеку яка містить алгоритмічек забезпечення для ПЗ;

– компонент UnitTests для тестування алгоритмічного забезпечення

– компонент WindowsFormApp, що відповідає за графічний інтерфейс користувача.

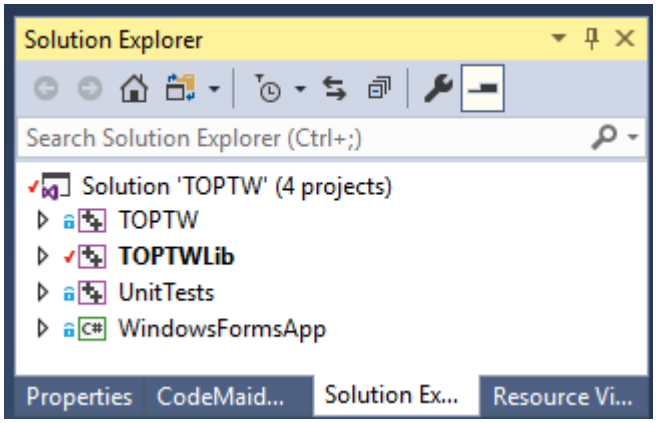


Рисунок 3.3 – Структура проекту ПЗ

Точкою входу є компонент WindowsFormApp. Коли користувач хоче ініціювати роботу алгоритму, то WindowsFormApp втикликає функції, що містятьться в динамічній бібліотеці TOPTW. При компілюванні і збірці TOPTW до неї включається весь вміст TOPTWLib. Тому, при виконанні WindowsFormApp потребує лише TOPTW бібліотеку. Компонент UnitTests тестує функціональність TOPTWLib.

3.4.1 Опис класів

Основна функціональність проекту сконцентрована в TOPTWLib. Вона реалізується засобами таких класів:

– AlgorithmClock;

– Node;

– InputData;

- ILS;
- NonParallelILS;
- InitialTemp;
- SAnnealing;
- VansteenwegenILS.

Клас AlgorithmClock призначений для вимірювання часу роботи функцій. Наприклад, для даного ПЗ він використовується для визначення часу роботи алгоритмів. Всі інші класи мають безпосереднє відношення до задачі TOPTW.

Вхідні дані задачі зберігаються у об’єкті класу InputData. Дані про кожну окрему вершину містяться в об’єктах класу Node.

Об’єкт класу InputData використовується алгоритмами: ILS, NonParallelILS, SAnnealing, VansteenwegenILS. Клас ILS відповідає за роботу модифікованого алгоритму повторюваного локального пошуку з паралельним визначенням околу, NonParallelILS – за роботу алгоритму повторюваного локальному з послідовним визначенням околу, SAnnealing – за роботу алгоритму імітаційного відпалу, VansteenwegenILS – за роботу алгоритму повторюваного локального пошуку з паралельним визначенням околу. Клас InitialTemp використовується щоб згенерувати початкову температуру для роботи алгоритму імітації відпалу.

У частині графічного метеріалу представлена схема структурна класів бібліотеки TOPTWLib.

Для візувлізації реалізовано такі основні класи в WindowsFormApp:

- CreateFileForm;
- MainMenu;
- DialogFm;
- RunAlgorithms;
- Texts.

Клас MainMenu описує об’єкти, що являють собою форми з головним меню користувача. Він являю собою макет для форм класу CreateFileForm і RunAlgorithms (дані форми наслідуються від MainMenu).

Форма для створення файлів реалізовується об'єктом класу CreateFileForm, форма для запуску алгоритмів – RunAlgorithms.

Об’єкти класу Texts містять ресурси, такі як повідомлення про неправильні вхідні дані.

Клас DialogFm дозволяє створювати об’єкти для відображення діалогового вікна з помилками.

3.4.2 Специфікація функцій

Специфікацію функцій компоненту TOPTW наведено в таблиці 3.1.

Таблиця 3.1 – Опис специфікації функцій бібліотеки TOPTW

Сигнатура функції	Опис вхідних параметрів	Опис вихідних параметрів	Призначення функції
extern "C" TOPTW_API BSTR RunAlgorithmsFromFiles (int firstAlgId, int secondAlgId, int maxTime, int routesNr, const char** files, int filesNr)	Шифр першого алгоритму; шифр другого алгоритму; Обмеження на час роботи; кількість маршрутів; файли для яких треба запустити алгоритми; кількість файлів	Назва файлу з результатами роботи алгоритмів	Запускає флгоритми з TOPTWLib

Специфікацію функцій для класів TOPTWLib наведено в таблицях 3.2 – 3.8.

Таблиця 3.2 – Опис специфікації функцій класу AlgorithmClock

Сигнатура функції	Опис вхідних параметрів	Опис вихідних параметрів	Призначення та/чи опис функції
AlgorithmClock()	-	-	Створює об’єкт класу AlgorithmClock
void StartOrReset()	-	-	Встановує початок відліку таймера на поточний час
double GetElapsedTimeFromStart()	-	Час, що пройшов з початку відліку	Викликається, коли необхідно визначити час між початком відліку таймера і поточним часом

Таблиця 3.3 – Опис специфікації функцій класу Node

Сигнатура функції	Опис вхідних параметрів	Опис вихідних параметрів	Призначення та/чи опис функції
Node()	-	-	Створює об’єкт, що являє собою вершину з пов’язаними до неї значеннями змінних
Node(Node * node, const int*const newNr)	Існуюча вершина; номер нової вершини	-	Створює нову вершину, для якої значення змінних копіюються з існуючої вершини
Node(int vertexNr,double xCoord,double yCoord,int visitingTime,int profit,int opening,int closing)	Номер вершини; координата по осі X; координата по осі У; час на відвідування вершини; корисність; час відкриття; останній момент початку відвідування	-	Створює нову вершину, значення змінних якої заповнюються на основі вхідних даних.
Node(ifstream * fstr)	Вхідний потік для роботи з файлами	-	Створює нову вершину, значення змінних якої зчитуються з файлу
int GetVertexNr()	-	Номер вершини	Повертає номер вершини
int GetVisitingTime()	-	Час відвідування	Повертає час, що необхідно витратити на відвідування даної вершини

Продовження таблиці 3.3

Сигнатура функції	Опис вхідних параметрів	Опис вихідних параметрів	Призначення та/чи опис функції
int GetProfit();	-	Корисність	Повертає корисність, що отримується при відвідуванні вершини
int GetOpeningTime();	-	Час відкриття	Повертає значення початку часового вікна для вершини
int GetClosingTime();	-	Час закриття	Повертає значення, що відповідає останньому моменту, коли можна почати відвідування вершини
void SetWaitTime(int wT);	Час очікування	-	Викликається, щоб встановити значення часу очікування перед відвідуванням вершини
int GetWaitTime()	-	Час очікування	Повертає значення часу очікування перед відвідуванням
void SetMaxShift(int mSht)	Час	-	Встановлює максимальний час на який допустимо відкласти відвідування вершини
int GetMaxShift()	-	Час	Повертає максимальний час на який допустимо відкласти відвідування вершини
double GetYCoord()	-	Координата	Повертає значення положення вершини по осі У
double GetXCoord()	-	Координата	Повертає значення положення вершини по осі Х
void SetArrivalTime (int arrT)	Момент прибуття	-	Встановлює значення часу прибуття в пункт
void SetVariables()	-	-	Встановлює початкові значення для змінних
int GetArrivalTime()	-	Момент прибуття	Повертає значення часу прибуття в пункт
int EuclDist(Node*)	Адреса вершини	Відстань	Розраховує евклідову відстань між двома вершинами

Таблиця 3.4 – Опис специфікації функцій класу InputData

Сигнатура функції	Опис вхідних параметрів	Опис вихідних параметрів	Призначення та/чи опис функції
InputData(int verticesNr, Node ** nodesArr)	Номер вершини; масив адрес вершин	-	На основі вхідних даних створює об’єкт , що містить необхідні вхідні дані для роботи алгоритмів
InputData(const char fPath[])	Повний шлях до файлу	-	Зчитує дані необхідні для роботи алгоритмів з файлу
Node **const GetVertices()	-	Вершина	Викликається щоб отримати положення масиву вершин
const int *const GetVerticesNr()	-	Кількість вершин	Викликається щоб отримати адресу змінної зі значенням кількості вершин
void ReadData()	-	-	Допоміжна функція для зчитування з файлу

Таблиця 3.5 – Опис специфікації функцій класу ILS

Сигнатура функції	Опис вхідних параметрів	Опис вихідних параметрів	Призначення та/чи опис функції
ILS(InputData *inputData, const int *rtNr,int nrOfShakes=5, bool stopOnTime = false, int maxTimeSec = 0)	Вхідні дні; кількість маршрутів; кількість ітерацій; критерій завершення; очікуваний час роботи	-	Створює об’єкт, що будує розв’язок задачі TOPTW за даними, що подаються на вхід. Встановлюється критерій виходу для ILS

Продовження таблиці 3.5

Сигнатура функції	Опис вхідних параметрів	Опис вихідних параметрів	Призначення та/чи опис функції
void Initialize()	-	-	Допоміжна функція для конструювання нового об’єкту
virtual void Run()	-	-	Запускає робуту модифікованого алгоритму повторюваного локального пошуку
Int GetFunctionValue()	-	Значення цільової функції	Повертає значення цільової функції після роботи алгоритму
void Shake()	-	-	Запускається для модифікації отриманого розв’язку
bool TimeElapsed (time_point startTm)	Час початку роботи алгоритму	Значення предикат	Викликається, щоб перевірити чи не перевищує час виконання алгоритму встановлену межу
int CalculateShift (int rIndx, int i, Node* insNd)	Номер маршруту; позиція в маршруті; адреса нової вершини	Збільшення тривалості маршруту спричинене додаванням нової вершини	Викликається щоб визначити величину на яку збільшиться тривалості маршруту через додаванням нової вершини в певну позицію
int EuclidDist (Node *v1, Node *v2)	Адреса вершини; адреса вершини	Евклідова відстань	Повертає збережену евклідову відстань між заданими вершинами

Продовження таблиці 3.5

Сигнатура функції	Опис вхідних параметрів	Опис вихідних параметрів	Призначення та/чи опис функції
virtual string ConsructedRoutesToString()	-	Побудовані маршрути	Повертає рядок, що містить послідовності вершин, що містять побудовані маршрути
void CalculateEuclidDistses()	-	-	Розраховує евклідові відстані між всіма вершинами і зберігає їх
virtual void InsertVertice (int routeIndx, int insPos,Node* insNd,int shift, int freeNdIndx)	Номер маршруту; позиція в маршруті; адреса нової вершини; зміна часу; позиція вершини	-	Викликається щоб додати нову вершину у маршрут в певну позицію
virtual void RemoveFromRoute (int * removePos, int removedVsNr, int m, bool saveHist = true)	Позиція у маршруті; кількість вершин;номер маршруту; показник збереження даних	-	Викликається щоб видалити певну кількість вершин на маршруті починаючи з визначеної позиціє. Дозволяє зберігати дані про те, які вершини і де було видалено.
void UpdateAfterRemoval (Node * leftVx, Node * rightVx,int rNdIndex)	Адреса вершини; адреса вершини; номер маршруту	-	Перераховує значення змінник після видалення вершин з маршруту.

Продовження таблиці 3.5

Сигнатура функції	Опис вхідних параметрів	Опис вихідних параметрів	Призначення та/чи опис функції
void ChangeShakeParams()	-	-	Регулює параметри збурення розв’язку
virtual void LS(double degree=2)	Степінь, що впливає на значимість корисності	-	Запускає алгоритм локального пошуку
void EvaluateNode (Node* prevNode, Node* crntNode)	Адреса вершини; адреса вершини; номер маршруту	-	Розраховує значення змінних, що була вставлена у маршрут
int PossibleWaitTime (Node *v1, Node *v2)	Адреса 1-ої вершини; адреса 2-ої вершини	Час очікування	Розраховує час очікування перед відвідуванням 2-ої вершини, при умові, що спочатку відвідується 1-а.
void EvaluateNode (Node* prevNode, Node* crntNode)	Адреса вершини; адреса вершини; номер маршруту	-	Розраховує значення змінних, що була вставлена у маршрут
void UpdateMaxShift (vector<Node *>& route, int NdPos)	Адреса маршруту; позиція в маршруті;	-	Оновлює змінну MaxShift для вставленої вершини і вершин, що передують їй

Продовження таблиці 3.5

Сигнатура функції	Опис вхідних параметрів	Опис вихідних параметрів	Призначення та/чи опис функції
int EvaluateSubsequentNodes (vector<Node*>*route, int insertedNdPos, int insertedNdShift)	Адреса маршруту; позиція в маршруті; зсув у часі	Позиція, де оновлення завершилось	Оновлює значення змінних для вершин, що слідують вставленій. Оновлення завершується, коли зустрічається вершина на яку вставка не вплинула
int EvaluateSubsequentNodesAfterRemoval (vector<Node *> *route, int crntNd, int shift)	Адреса маршруту; адреса вершини; час на який змінилась тривалість маршруту	Позиція вершини, де оновлення завершилось	Оновлює значення змінних для вершин, що слідують видаленій(им). Оновлення завершується, коли зустрічається вершина на яку зміни не вплинули

Таблиця 3.6 – Опис специфікації функцій класу InitialTemp

Сигнатура функції	Опис вхідних параметрів	Опис вихідних параметрів	Призначення та/чи опис функції
InitialTemp (SAnnealing* ils)	адреса об’єкта, що відповідає з алгоритм імітації відпалу	-	Створення об’єкта класу InitialTemp
double CalculateInitialTemp()	-	Початкова температура	Повертає значення початкової температури
Void GenerateSolutions()	-	-	Будує множину початкових розв’язків, модифікує їх і генерує нові розв’язки.

Продовження таблиці 3.6

Сигнатура функції	Опис вхідних параметрів	Опис вихідних параметрів	Призначення та/чи опис функції
int CalculateEnergyDiff()	-	Різниця між значеннями цільової функції	Генерує розв’язок, модифікує його і знаходить новий розв’язок. Розраховує різницю між значеннями цільової функції отриманих розв’язків
bool CalculateRmPos (int &rmPos, SAnnealing * alg)	Адреса позиції для видалення; адреса об’єкта, що виконує алгоритм імітаційного відпалу	Предикат позначає можливість подальшого видалення вершин	Викликається, коли необхідно розрахувати позицію для видалення вершин(и) при процедурі збурення розв’язку

Таблиця 3.7 – Опис специфікації функцій класу InitialTemp

Сигнатура функції	Опис вхідних параметрів	Опис вихідних параметрів	Призначення та/чи опис функції
SAnnealing (InputData *inputData, const int *rtNr,double InitialT=1, double alfa = 0.95, int itrNr = 5)	Вхідні дані; кількість иаршрутів; початкова температура; Константа для геометричного закону; максимальна кількість ітерацій	-	Створення об’єкта класу SAnnealing на основі вхідних даних. Ініціалізація вхідних даних, що необхідні для роботи алгоритму імітаційного відпалу
virtual void Run() override	-	-	Викликається щоб запустити роботу алгоритму імітаційного відпалу

Таблиця 3.8 – Опис специфікації функцій класу VansteenwegenILS

Сигнатура функції	Опис вхідних параметрів	Опис вихідних параметрів	Призначення та/чи опис функції
VansteenwegenILS (InputData *inputData, const int *rtNr, int nrOfShakes = 5, bool stopOnTime = false, int maxTimeSec = 0)	Дані про топологію; кількість маршрутів; наявність ліміту часу; величина ліміту	-	Створення об’єкта класу VansteenwegenILS на основі вхідних даних. Ініціалізація вхідних даних, що необхідні для роботи алгоритму повторюваного локального пошуку
Void ChangeShakeParams()	-	-	Оновлює кількість видалених вершин і позицію для видалення після кожної ітерації
virtual void Run() override	-	-	Викликається щоб запустити роботу алгоритму повторюваного локального пошуку

3.5 Висновки до розділу

Даний розділ присвячено опису програмного забезпечення, в ньому наведено опис вхідних та вихідних даних, надано обґрунтування вибору засобів розробки і їх основні характеристики. Описано архітектуру програмного забезпечення на рівні компонентів і класів. Надано мотивіцію зробленим архітектурним рішенням. Наведено специфікацію функцій.

4 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

4.1 Вимоги до технічного забезпечення

Для роботи застосування необхідно мати встановлений .NET Framework 4.6.1, всі інші вимоги до технічного забезпечення наведені у таблиці 4.1.

Необхідною умовою також є те, що виконуваний файл і динамічна бібліотека TOPTW.dll повинні знаходитись в одній папці з ПЗ.

Таблиця 4.1 – Конфігурація технічного забезпечення

Процесор	233 MHz та більше
Операційна система	Windows NT
Оперативна пам’ять	64 Мб RAM та більше
Відеоадаптер і монітор	VGA (640 x 480) та вище
Вільне місце на диску	450 Кб мінімум
Пристрої взаємодії з користувачем	Клавіатура і миша

4.2 Керівництво користувача

Екранні форми, які може використовувати користувач для роботи з ПЗ представлено в частині графічного матеріалу. На кожній формі присутнє меню для навігації користувача. Воно розташоване зверху.

Для того, щоб створити файл з вхідними даними, який потім може бути використаний при роботі алгоритмів, користувачу треба обрати вкладку меню «Створити файл». Вигляд форми для роботи з файлами наведено рисунку 4.1.

На формі присутня таблиця в якій відображаються дані про вершини, що будуть входити в топологію (граф). Щоб додати нову вершину в неї, треба заповнити поля над таблицею і натиснути кнопку «Додати пункт». Після того як користувач заповнює поля і натискає кнопку «Додати пункт» система перевіряє введені дані і у випадку їх незадовільності виводить користувачу повідомлення. Якщо користувач не заповнює хоча б одне з полів, йому відображається вікно з повідомленням «При створенні нового пункту всі атрибути(поля) повинні мати значення». Після того як користувач заповнює всі поля, перевіряється чи введений

час відкриття не пізніше за час закриття. Ситуація, коли дана умова порушується, зображена на рисунку 4.2.

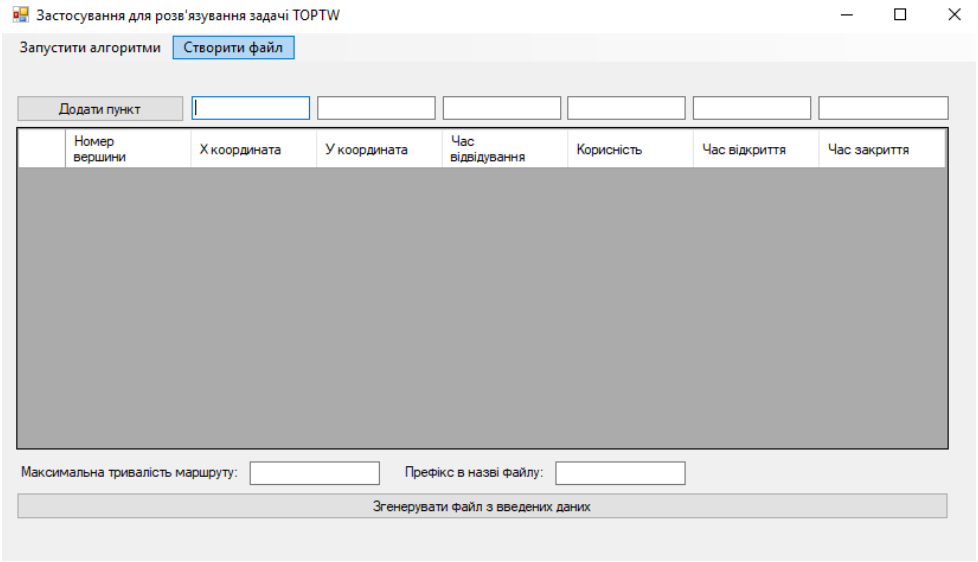


Рисунок 4.1 – Форма для створення файлів

Користувачу необхідно контрулювати формат введених даних. Всі дані у полях, що зображені на рисунку 4.2 повинні бути числовими. Всі данні окрім координат – цілочислові. У випадку, коли користувач порушує дане обмеження, йому виведеться діалогове вікно з попередженням про порушення формату введених даних. Дана ситуація зображена на рисунку 4.3.

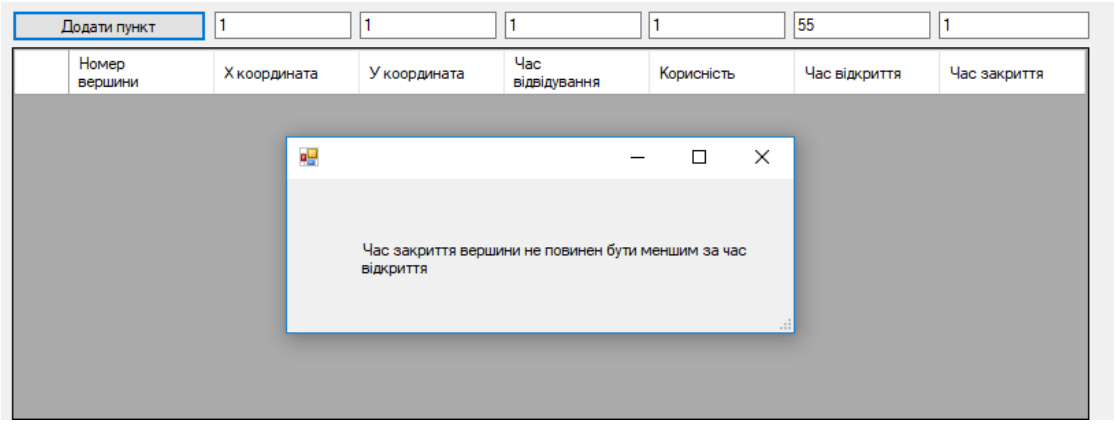


Рисунок 4.2 – Порушення обмежень встановлених для часових вікон вершин

Коли користувач вводить коректні вхідні дані і натискає кнопку «Додати пункт», новий запис з’являється в таблиці. Для того щоб його видалити, треба його виділити і на клавіатурі натиснути кнопку «Delete». Це відображено на рисунку 4.4. Рядки (вершини) в таблиці можна редагувати. Для цього необхідно два рази натиснути на комірку таблиці, яку треба відредагувати. Приклад наведено на

рисунку 4.5. Редагувати можна всі колонки окрім колонки «Номер вершини». Якщо користувач вводитиме незадовільні дані, то йому будить виводитись повідомлення, як і при додаванні вершини.

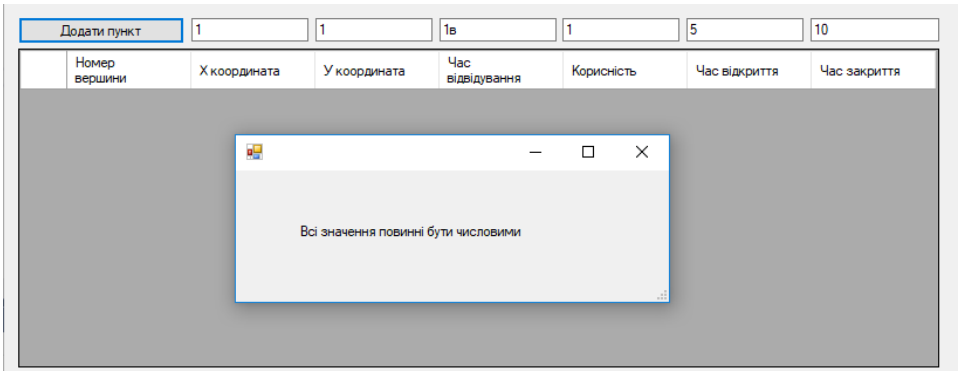


Рисунок 4.3 – Незадовільний формат даних для атрибутів вершини

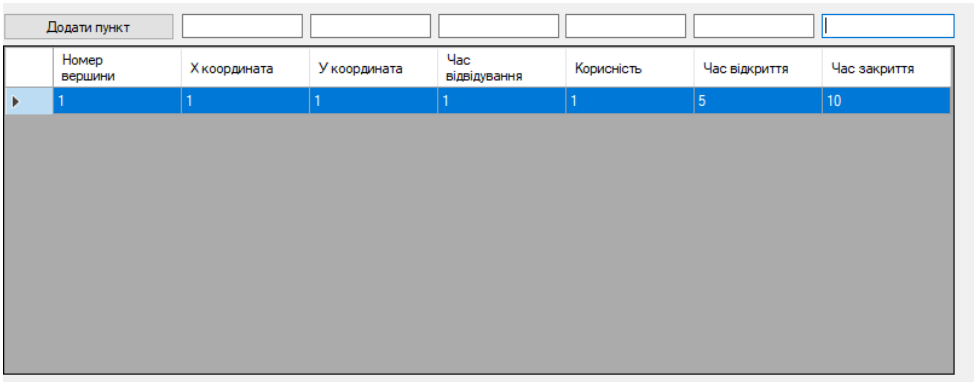


Рисунок 4.4 – Виділення нової вершини

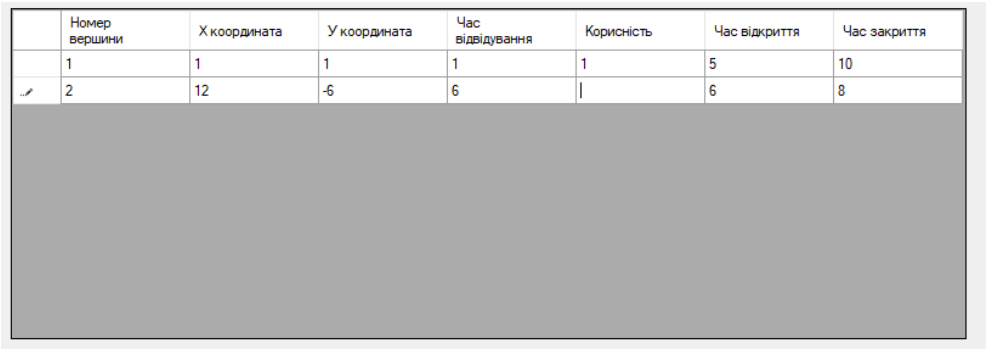


Рисунок 4.5 – Редагування вершини

Коли користувач вводить дані про всі вершини, що формують топологію для задачі, йому необхідно заповнити ще два поля, а саме, всказати максимальну тривалість маршрутів і префікс на основі якого буде названо файл.

Щоб сформувати файл вхідних даних для задачі TOPTW, користувач натискає кнопку «Згенерувати файл з наведених даних» (див. рис. 4.1). Якщо не введено жодної вершини, чи не задано префікс, чи обмеження на час маршруту, користувачу

виведеться повідомлення про це. В іншому випадку сформується файл в папці з застосуванням. Його буде названо згідно з наведеним префіксом і кількістю введених в таблицю вершин.

Щоб запустити алгоритми користувачу необхідно натиснути на пункт меню «Запустити алгоритми». Форма для запуску алгоритмів наведена на рисунку 4.6.

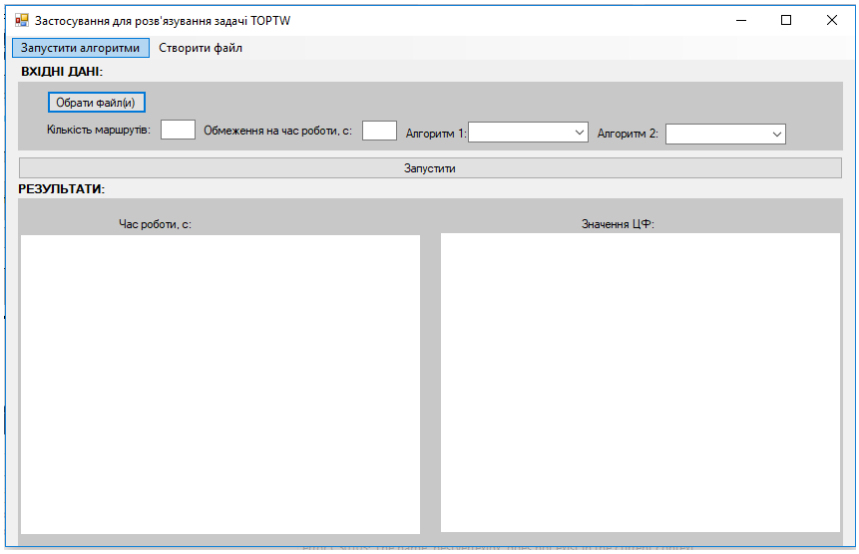


Рисунок 4.6 – Форма для запуску алгоритмів

Перед тим як запускати алгоритми, необхідно заповнити поля в блоці «Вхідні дані». Щоб обрати файли вхідних даних для роботи алгоритмів треба натиснути кнопку «Обрати файли(м)». Відкриється діалогове вікно за допомогою якого можна обрати потрібні файли. Після того, як файли буде обрано, з’явиться кнопка «Відобразити список файлів». Натиснувши на неї, можна переглянути список файлів, що було обрано. Приклад наведено на рисунку 4.7. Для кожного з цих файлів буде запущено обраний(і) алгоритм(и).

Необхідно задати кількість маршрутів, обмеження на час роботи алгоритму(ів) і з випадаючого списку(ів) обрати алгоритм(и). Після цього треба натиснути кнопку «Запустити». У випадку, коли не обрано алгоритм для жодного з випадаючих списків, з’явиться повідомлення, що зображено на рисунку 4.8.

В такому ж вікні відображатимуться повідомлення «Введіть кількість маршрутів перед запуском», «Введіть часове обмеження для роботи алгоритму в секунда», «Будь ласка, оберіть різні алгоритми перед запуском». Щоб уникнути

подібних повідомлень, потрібно ввести коректні дані для роботи алгоритмів і вже тоді натиснути кнопку «Запустити».

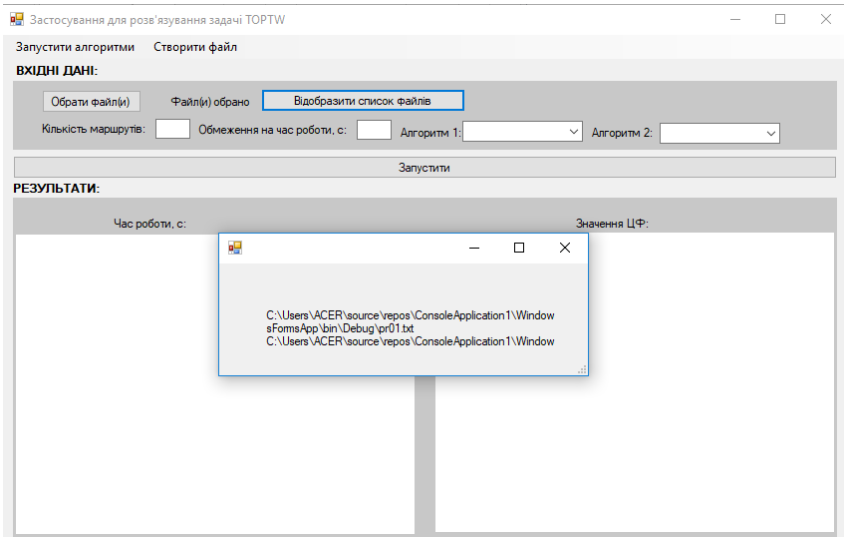


Рисунок 4.7 – Вибір файлів для запуску

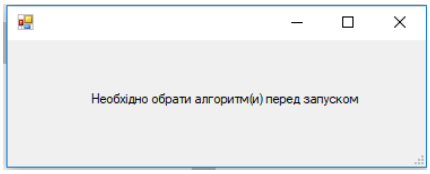


Рисунок 4.8 – Повідомлення про відсутність обраних алгоритмів

В результаті запуску алгоритмів формується файл з результатами, час роботи алгоритмів і отримані значення цільової функціїї відображаються на графіках. Приклад роботи алгоритмів наведено на рисунку 4.9.

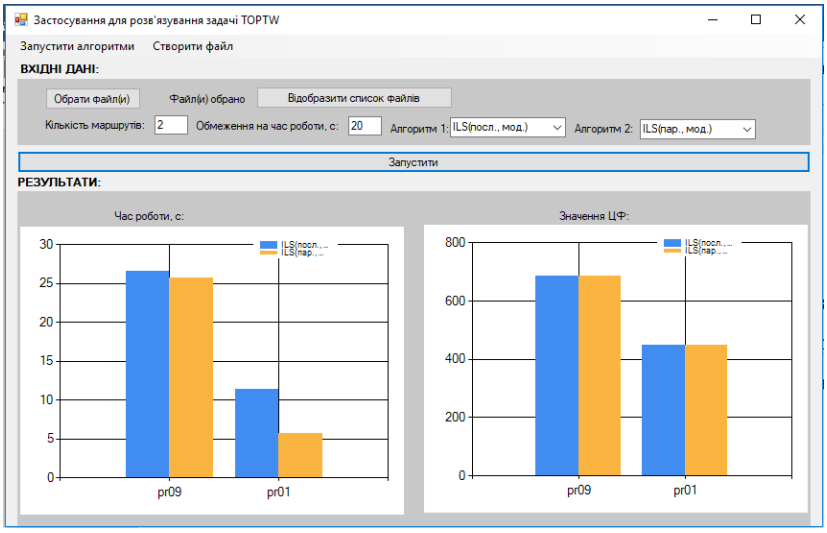


Рисунок 4.9 – Візуалізаці роботи алгоритмів

4.3 Висновки до розділу

Даний розділ присвячено опису процесу роботи з ПЗ. Наведено список мінімальних вимог для роботи з ПЗ. Описано процес роботи з ПЗ.

5 РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ

5.1 Дослідження ефективності використаних алгоритмів

Метою досліджень є оцінка ефективності розроблених алгоритмів та пропонованих модифікацій для розв'язування задачі TOPTW; порівняння алгоритмів, виявлення їх недоліків і переваг.

Використовувалось таке апаратне забезпечення: процесор Intel Core i5-3317U, 1.7 GHz з оперативною пам`яттю 4 GB.

Для дослідження ефективності запропонованого алгоритму методу повторюваного локального пошуку проводились експерименти на 27 наборах даних, взятих із [63]. Будувалось три маршрути з часовим обмеженням на роботу алгоритмів у 40 секунд. З такими вхідними даними було запущено модифікований і немодифікований алгоритми локального пошуку, результати наведені у частині графічного матеріалу.

Далі проводилось дослідження ефективності модифікованого алгоритму повторюваного локального пошуку та імітаційного відпалу. Використовувались набори даних сконструйовані на основі наборів для VRP [63]. Результати експериментів наведено в таблиці 5.1. Зафарбовані комірки показують, що алгоритм ILS завершив свою роботу раніше виділеного часу.

Таблиця 5.1 – Результати виконання алгоритмів ILS та SA

Назва файлу	Кількість вершин	Час роботи ILS, с	Значення ЦФ ILS	Похибка ILS,%	Час роботи SA , с	Знач. ЦФ SA	Похибка SA, %	Краще відоме знач. ЦФ	Різниця похибок, %
Кількість маршрутів m=2, обмеження часу виконання t=10c									
c101	100	5.20	570	3.39	10.65	570	3.39	590	0.00
c102	100	5.66	640	1.54	10.30	640	1.54	650	0.00
c103	100	10.71	680	4.23	10.42	650	8.45	710	4.23
c104	100	9.10	700	4.11	10.57	730	0.00	730	-4.11
c105	100	7.93	640	0.00	10.59	640	0.00	640	0.00
c106	100	7.15	620	0.00	10.51	620	0.00	620	0.00
c107	100	10.52	660	0.00	10.52	660	0.00	660	0.00

Продовження таблиці 5.1

Назва файлу	Кількість вершин	Час роботи ILS, с	Значення ЦФ ILS	Похибка ILS,%	Час роботи SA , с	Знач. ЦФ SA	Похибка SA, %	Краще відоме знач. ЦФ	Різниця похибок, %
Кількість маршрутів m=2, обмеження часу виконання t=10с									
c108	100	10.01	670	1.47	10.51	670	1.47	680	0.00
c109	100	10.63	690	2.82	10.52	690	2.82	710	0.00
pr01	48	3.83	450	7.02	10.11	442	8.68	484	1.65
pr02	96	11.48	640	6.16	10.52	640	6.16	682	0.00
pr03	144	10.96	643	10.45	10.69	642	10.58	718	0.14
pr04	192	13.20	695	20.21	12.19	708	18.71	871	-1.49
pr05	240	12.91	859	16.03	12.80	859	16.03	1023	0.00
pr06	288	14.16	852	13.06	14.64	852	13.06	980	0.00
pr07	72	10.88	523	6.61	10.60	517	7.68	560	1.07
pr08	144	12.23	644	20.49	11.95	644	20.49	810	0.00
pr09	216	13.20	646	22.82	13.47	646	22.82	837	0.00
r101	100	6.34	340	2.58	10.96	340	2.58	349	0.00
r102	100	12.15	494	2.76	10.40	494	2.76	508	0.00
r103	100	9.26	513	0.77	11.03	513	0.77	517	0.00
r104	100	11.59	511	5.55	10.71	511	5.55	541	0.00
r105	100	11.65	472	-4.19	10.81	449	0.88	453	5.08
r106	100	11.43	522	0.57	11.06	522	0.57	525	0.00
r107	100	8.08	519	2.99	10.55	519	2.99	535	0.00
r108	100	12.92	526	5.73	11.21	526	5.73	558	0.00
r109	100	10.56	484	2.81	11.20	484	2.81	498	0.00
Кількість маршрутів m=3, обмеження часу виконання t =10с									
c101	100	8.04526	780	3.70	12.0661	780	3.70	810	0.00
c102	100	8.97144	870	3.33	11.0513	860	4.44	900	1.11
c103	100	11.0706	940	2.08	10.3933	940	2.08	960	0.00
c104	100	10.8448	940	5.05	10.712	950	4.04	990	-1.01
c105	100	10.6826	840	2.33	11.5304	810	5.81	860	3.49
c106	100	11.1701	840	2.33	10.3909	830	3.49	860	1.16
c107	100	10.519	850	4.49	10.6453	850	4.49	890	0.00
c108	100	10.4625	890	1.11	11.2609	890	1.11	900	0.00

Продовження таблиці 5.1

Назва файлу	Кількість вершин	Час роботи ILS, с	Значення ЦФ ILS	Похибка ILS,%	Час роботи SA , с	Знач. ЦФ SA	Похибка SA, %	Краще відоме знач. ЦФ	Різниця похибок, %
Кількість маршрутів m=3, обмеження часу виконання t =10c									
c109	100	10.442	920	2.13	10.4874	900	4.26	940	2.13
pr01	48	3.99206	580	3.65	10.0942	580	3.65	602	0.00
pr02	96	11.9542	823	7.94	10.7276	823	7.94	894	0.00
pr03	144	12.9435	842	12.56	13.7032	842	12.56	963	0.00
pr04	192	17.7269	1066	11.17	20.2806	1066	11.17	1200	0.00
pr05	240	23.2442	1244	8.80	25.166	1244	8.80	1364	0.00
pr06	288	21.5724	1177	13.58	25.0679	1177	13.58	1362	0.00
pr07	72	10.304	698	3.72	10.5029	677	6.62	725	2.90
pr08	144	11.4885	947	11.82	15.3713	947	11.82	1074	0.00
pr09	216	19.0642	1019	13.28	20.5016	1019	13.28	1175	0.00
r101	100	2.38457	438	9.50	11.1031	438	9.50	484	0.00
r102	100	10.5227	646	5.83	10.7848	646	5.83	686	0.00
r103	100	11.1175	693	5.07	11.085	693	5.07	730	0.00
r104	100	10.7632	702	7.75	10.4514	702	7.75	761	0.00
r105	100	8.88443	631	-1.94	10.546	631	-1.94	619	0.00
r106	100	11.1931	728	-1.82	11.1987	674	5.73	715	7.55
r107	100	12.1705	692	7.11	11.2187	692	7.11	745	0.00
r108	100	12.459	724	6.94	10.8471	724	6.94	778	0.00
r109	100	12.2556	634	9.69	10.3406	625	10.97	702	1.28
Кількість маршрутів m=2, обмеження часу виконання t =20c									
c101	100	4.98194	570	3.39	20.291	570	3.39	590	0.00
c102	100	5.47878	640	1.54	20.3962	630	3.08	650	1.54
c103	100	12.3183	680	4.23	20.3081	660	7.04	710	2.82
c104	100	8.34995	700	4.11	20.8211	700	4.11	730	0.00
c105	100	7.26274	640	0.00	20.4093	640	0.00	640	0.00
c106	100	9.20604	620	0.00	20.5463	620	0.00	620	0.00
c107	100	14.1443	660	0.00	20.4715	630	4.55	660	4.55
c108	100	11.8131	670	1.47	20.3337	670	1.47	680	0.00
c109	100	10.7699	690	2.82	20.8828	700	1.41	710	-1.41

Продовження таблиці 5.1

Назва файлу	Кількість вершин	Час роботи ILS, с	Значення ЦФ ILS	Похибка ILS,%	Час роботи SA , с	Знач. ЦФ SA	Похибка SA, %	Краще відоме знач. ЦФ	Різниця похибок, %
Кількість маршрутів m=2, обмеження часу виконання t =20с									
pr01	48	4.99885	450	7.02	20.1142	438	9.50	484	2.48
pr02	96	22.59	655	3.96	21.5911	652	4.40	682	0.44
pr03	144	21.8718	649	9.61	20.6946	643	10.45	718	0.84
pr04	192	25.3429	722	17.11	21.3924	707	18.83	871	1.72
pr05	240	26.62	859	16.03	28.101	859	16.03	1023	0.00
pr06	288	31.2305	852	13.06	31.1972	852	13.06	980	0.00
pr07	72	15.4457	523	6.61	21.2709	523	6.61	560	0.00
pr08	144	21.3254	653	19.38	22.3905	654	19.26	810	-0.12
pr09	216	23.7174	686	18.04	26.9255	646	22.82	837	4.78
r101	100	3.43922	340	2.58	21.4911	340	2.58	349	0.00
r102	100	13.656	494	2.76	20.5374	489	3.74	508	0.98
r103	100	15.0161	513	0.77	21.0423	513	0.77	517	0.00
r104	100	15.6933	511	5.55	21.5526	511	5.55	541	0.00
r105	100	14.5062	478	-5.52	20.9089	412	9.05	453	14.57
r106	100	11.4099	522	0.57	21.1925	522	0.57	525	0.00
r107	100	18.7168	519	2.99	20.517	527	1.50	535	-1.50
r108	100	19.297	526	5.73	20.3915	526	5.73	558	0.00
r109	100	16.601	484	2.81	20.469	427	14.26	498	11.45
Кількість маршрутів m=3, обмеження часу виконання t =20с									
c101	100	9.07432	780	3.70	20.5093	780	3.70	810	0.00
c102	100	11.2409	870	3.33	20.4902	880	2.22	900	-1.11
c103	100	18.6001	940	2.08	20.2895	920	4.17	960	2.08
c104	100	19.0346	940	5.05	20.8657	960	3.03	990	-2.02
c105	100	15.4703	840	2.33	21.2214	840	2.33	860	0.00
c106	100	16.7319	840	2.33	21.2151	830	3.49	860	1.16
c107	100	20.3872	860	3.37	20.8258	860	3.37	890	0.00
c108	100	21.2946	890	1.11	20.3831	890	1.11	900	0.00
c109	100	21.5562	920	2.13	20.7299	930	1.06	940	-1.06

Продовження таблиці 5.1

Назва файлу	Кількість вершин	Час роботи ILS, с	Значення ЦФ ILS	Похибка ILS,%	Час роботи SA , с	Знач. ЦФ SA	Похибка SA, %	Краще відоме знач. ЦФ	Різниця похибок, %
Кількість маршрутів m=3, обмеження часу виконання t =20с									
pr01	48	5.79311	580	3.65	20.219	577	4.15	602	0.50
pr02	96	22.364	868	2.91	20.8547	843	5.70	894	2.80
pr03	144	22.6746	842	12.56	23.3112	842	12.56	963	0.00
pr04	192	22.0462	1066	11.17	22.7909	1078	10.17	1200	-1.00
pr05	240	41.1855	1244	8.80	36.5613	1244	8.80	1364	0.00
pr06	288	42.2009	1177	13.58	53.5806	1177	13.58	1362	0.00
pr07	72	20.3966	709	2.21	20.6886	667	8.00	725	5.79
pr08	144	23.3895	947	11.82	20.7222	948	11.73	1074	-0.09
pr09	216	29.3671	1019	13.28	34.7676	1019	13.28	1175	0.00
r101	100	5.66315	438	9.50	20.7263	464	4.13	484	-5.37
r102	100	17.8757	646	5.83	21.0709	646	5.83	686	0.00
r103	100	21.0006	693	5.07	20.6237	704	3.56	730	-1.51
r104	100	20.5995	702	7.75	20.4109	707	7.10	761	-0.66
r105	100	12.4442	631	-1.94	20.7205	562	9.21	619	11.15
r106	100	21.4909	757	-5.87	21.2239	678	5.17	715	11.05
r107	100	17.1512	692	7.11	20.9595	711	4.56	745	-2.55
r108	100	20.8815	724	6.94	20.3157	731	6.04	778	-0.90
r109	100	21.0063	637	9.26	21.4826	623	11.25	702	1.99
Кількість маршрутів m=2, обмеження часу виконання t =40с									
c101	100	4.9708	570	3.39	41.25	570	3.39	590	0.00
c102	100	5.97947	640	1.54	40.62	630	3.08	650	1.54
c103	100	14.4166	680	4.23	40.88	710	0.00	710	-4.23
c104	100	9.95855	700	4.11	41.25	730	0.00	730	-4.11
c105	100	7.61226	640	0.00	40.74	640	0.00	640	0.00
c106	100	9.20955	620	0.00	40.36	620	0.00	620	0.00
c107	100	13.5904	660	0.00	40.56	660	0.00	660	0.00
c108	100	11.4933	670	1.47	40.34	670	1.47	680	0.00
c109	100	12.0195	690	2.82	40.81	690	2.82	710	0.00

Продовження таблиці 5.1

Назва файлу	Кількість вершин	Час роботи ILS, с	Значення ЦФ ILS	Похибка ILS,%	Час роботи SA , с	Знач. ЦФ SA	Похибка SA, %	Краще відоме знач. ЦФ	Різниця похибок, %
Кількість маршрутів m=2, обмеження часу виконання t =40с									
pr01	48	4.84563	450	7.02	40.21	484	0.00	484	-7.02
pr02	96	41.0094	676	0.88	41.03	682	0.00	682	-0.88
pr03	144	41.6802	680	5.29	41.22	687	4.32	718	-0.97
pr04	192	41.6143	738	15.27	43.40	763	12.40	871	-2.87
pr05	240	42.7625	893	12.71	42.86	896	12.41	1023	-0.29
pr06	288	43.2829	860	12.24	43.86	861	12.14	980	-0.10
pr07	72	12.2225	523	6.61	40.54	560	0.00	560	-6.61
pr08	144	42.6629	695	14.20	41.32	716	11.60	810	-2.59
pr09	216	43.1534	697	16.73	42.28	655	21.74	837	5.02
r101	100	2.69258	340	2.58	40.39	340	2.58	349	0.00
r102	100	11.5199	494	2.76	40.50	494	2.76	508	0.00
r103	100	12.8544	513	0.77	40.56	513	0.77	517	0.00
r104	100	15.3709	511	5.55	40.57	511	5.55	541	0.00
r105	100	15.2195	478	-5.52	41.40	453	0.00	453	5.52
r106	100	11.7993	522	0.57	40.65	522	0.57	525	0.00
r107	100	14.9367	519	2.99	40.86	519	2.99	535	0.00
r108	100	18.2739	526	5.73	40.65	526	5.73	558	0.00
r109	100	15.1	484	2.81	40.57	472	5.22	498	2.41
Кількість маршрутів m=3, обмеження часу виконання t =40с									
c101	100	10.9885	780	3.70	43.14	780	3.70	810	0.00
c102	100	17.9966	870	3.33	41.64	890	1.11	900	-2.22
c103	100	29.6285	940	2.08	42.61	940	2.08	960	0.00
c104	100	24.3113	940	5.05	43.92	970	2.02	990	-3.03
c105	100	15.1197	840	2.33	41.45	840	2.33	860	0.00
c106	100	15.1272	840	2.33	41.36	820	4.65	860	2.33
c107	100	38.8412	890	0.00	40.65	880	1.12	890	1.12
c108	100	19.3506	890	1.11	42.49	890	1.11	900	0.00
c109	100	38.1773	940	0.00	44.65	940	0.00	940	0.00

Продовження таблиці 5.1

Назва файлу	Кількість вершин	Час роботи ILS, с	Значення ЦФ ILS	Похибка ILS,%	Час роботи SA , с	Знач. ЦФ SA	Похибка SA, %	Краще відоме знач. ЦФ	Різниця похибок, %
Кількість маршрутів m=3, обмеження часу виконання t =40c									
pr01	48	5.2616	580	3.65	40.52	580	3.65	602	0.00
pr02	96	43.6293	878	1.79	45.91	849	5.03	894	3.24
pr03	144	41.3597	891	7.48	49.71	841	12.67	963	5.19
pr04	192	44.5807	1093	8.92	47.61	1078	10.17	1200	1.25
pr05	240	45.5188	1244	8.80	76.07	1244	8.80	1364	0.00
pr06	288	47.6005	1177	13.58	77.76	1177	13.58	1362	0.00
pr07	72	25.8317	721	0.55	41.85	676	6.76	725	6.21
pr08	144	41.7968	961	10.52	47.79	961	10.52	1074	0.00
pr09	216	45.9012	1095	6.81	43.20	1019	13.28	1175	6.47
r101	100	4.60142	438	9.50	42.60	438	9.50	484	0.00
r102	100	19.3809	646	5.83	41.96	646	5.83	686	0.00
r103	100	19.2088	693	5.07	41.96	693	5.07	730	0.00
r104	100	22.6753	702	7.75	43.24	702	7.75	761	0.00
r105	100	11.2694	631	-1.94	41.60	556	10.18	619	12.12
r106	100	24.0777	762	-6.57	40.99	683	4.48	715	11.05
r107	100	19.565	692	7.11	42.73	692	7.11	745	0.00
r108	100	23.4561	724	6.94	40.83	724	6.94	778	0.00
r109	100	21.6781	637	9.26	41.82	639	8.97	702	-0.28

Було проведено 162 прогони для кожного з алгоритмів. Було використано 27 наборів вхідних даних на яких конструювалось $m \in \{2,3\}$ маршрути. При кожному запуску задавалось обмеження на час роботи алгоритму $time \in \{10, 20, 40\}$. Наступна ітерація алгоритму не починається, якщо алгоритм виконується довше, ніж time. В результаті, для 44 прогонів алгоритм ILS знаходив кращі розв'язки ніж SA, що складає 27% всіх прогонів. Для 29 прогонів, навпаки, SA давав кращі результати, що складає 18% від усіх прогонів. В таблиці 5.2 наведено результати порівняння ефективності роботи алгоритмів для різних використаних конфігурацій.

Якщо в комірці таблиці наведено назву алгоритму, то це означає, що даний алгоритм для даної конфігурації згенерував розв’язок з кращим значенням цільової функцій. Кольором виділені комірки, де алгоритм ILS завершив роботу раніше виділеного часу, іншими словами, знайшов локальний оптимум в множині розв’язків, що генерує локальний пошук.

Таблиця 5.2 – Порівняння ефективності роботи алгоритмів

	m=2				m=3		
	time=10s	time=20s	time=40s		time=10s	time=20s	time=40s
c101				c101			
c102		ILS	ILS	c102	ILS	SA	SA
c103	ILS	ILS	SA	c103		ILS	
c104	SA		SA	c104	SA	SA	SA
c105				c105	ILS		
c106				c106	ILS	ILS	ILS
c107		ILS		c107			ILS
c108				c108			
c109		SA		c109	ILS	SA	
pr01	ILS	ILS	SA	pr01		ILS	
pr02		ILS	SA	pr02		ILS	ILS
pr03	ILS	ILS	SA	pr03			ILS
pr04	SA	ILS	SA	pr04		SA	ILS
pr05			SA	pr05			
pr06			SA	pr06			
pr07	ILS		SA	pr07	ILS	ILS	ILS
pr08		SA	SA	pr08		SA	
pr09		ILS	ILS	pr09			ILS
r101				r101		SA	
r102		ILS		r102			
r103				r103		SA	
r104				r104		SA	
r105		ILS	ILS	r105		ILS	ILS
r106				r106		ILS	ILS
r107		SA		r107		SA	
r108				r108		SA	
r109		ILS	ILS	r109		ILS	SA

В результаті, загальний час роботи алгоритму ILS на всіх прогонах склав 2774 с, коли для SA цей показник складає 4200 с. Хоч обмеження на час виконання алгоритмів задавались однакові, алгоритм ILS в багатьох випадках закінчує свою роботу раніше. Наприклад, при побудові 2 маршрутів і обмеженні у 10с співвідношення часу роботи алгоритмів показано на рисунку 5.1.

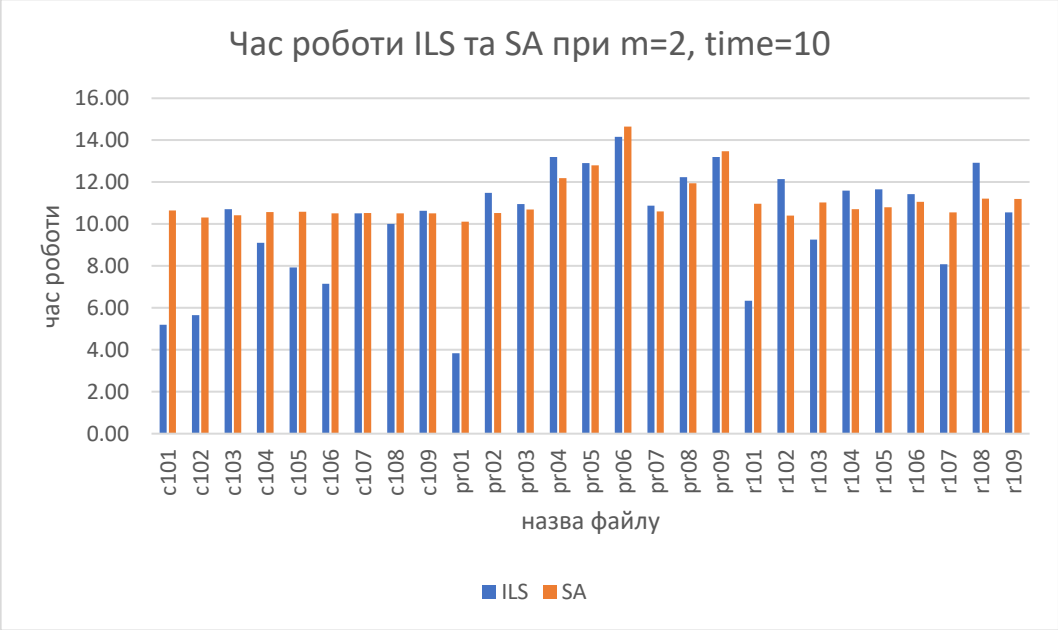


Рисунок 5.1 – Співвідношення часу роботи алгоритмів ILS та SA

В основу даних алгоритмів лягла процедура локального пошуку з паралельним обчисленням околу. Щоб оцінити зміну часу роботи порівняно з послідовним обчисленням околу було проведено експерименти на наборах даних pr01- pr09.

Випробування відбувались з такими параметрами:

– будується 2 маршрути;

– виконується 20 ітерацій алгоритму.

Для кожного набору даних проводилось по 5 прогонів з паралельним і послідовним обчисленням околу. Усереднені результати наведено в таблиці 5.3. На рисунку 5.2 наведено навантаження на процесори при роботі алгоритму ILS з використанням послідовного ЛП, а на рисунку 5.3 наведено роботу ILS з паралельним ЛП.

Таблиця 5.3 – Усереднений час роботи алгоритму повторюваного ЛП

Кількість вершин	48	72	96	144	160	192	216	240	288
Час роботи послідовного ЛП, с	4.97	21.45	27.8	47.72	57.77	78.72	89.272	156.9	167.48
Час роботи паралельного ЛП, с	3.23	12.3	21.03	39.56	44.43	69.33	70.8	108.8	127.7
Покращення, %	35	42.6	24.34	17.09	23.09	11.92	20.69	30.61	23.75

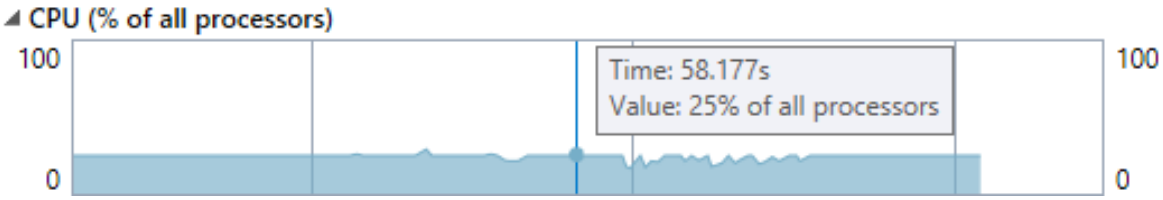


Рисунок 5.2 – Завантаженість процесорів при роботі послідовного алгоритму ЛП

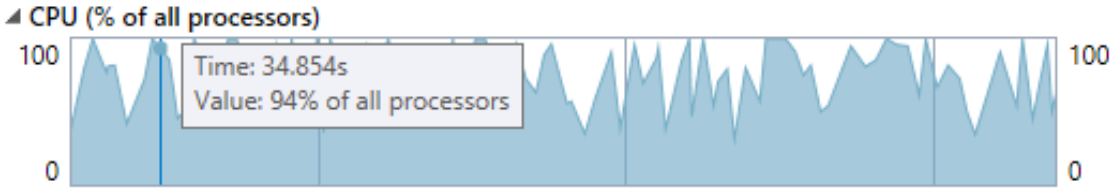


Рисунок 5.3 – Завантаженість процесорів при роботі паралельного алгоритму ЛП

В частині графічного матеріалу наведено порівняння роботи послідовної і паралельної реалізацій модифікованого ILS при зміні конфігурації технічного забезпечення на таку: 2.2GHz, Intel Core i7, 16GB RAM.

5.2 Аналіз отриманих результатів

Для проведених прогонів модифікованого повторюваного локального пошуку загальний час роботи складає 716 с, а для немодифікованого – 1101 с, що на 35% довше. Хоч алгоритм модифікованого повторюваного локального пошуку не приносить результатів у напрямку покращення якості знайденого розв’язку, проте час роботи скорочується. Це досягається тому, що модифікований алгоритм має

високу ступінь інтенсифікації пошуку у просторі розв’язків ЛП, в той час як для оригінального алгоритму характер пошукув просторі розв’язків ЛП ближче до випадкового.

З огляду на це, доречно використовувати таку модифікацію. Далі модифікований ILS порівнюється з SA.

В основі алгоритмів SA та ILS лежить однакова процедура локального пошуку. Судячи з результатів в таблиці 5.3, час роботи для паралельної реалізації у середньому на 25% краще за послідовну версію. Крім того, з рисунків 5.1, 5.2 видно, що балансування навантаження на процесори краще.

Тож дану модифікацію доцільно запровадити для подальшої роботи алгоритмів. Крім того, дане покращення у швидкодії буде збільшуватись зі збільшенням кількості задіяних процессорів, але ідеальної завнтаженості процессорів для алгоритмів досягнути не вдасться, адже велика доля обчислень у SA та ILS виконується послідовно. Варто зазначити, що коли кількість процесорів зростатиме більше ніж n/m, то пришвидшення не зростатиме, бо процесори не будуть використані при обчисленні околу (n – кількість вершин, $m-$ кількість маршрутів).

В результаті проведених експериментів, видно, що основною перевагою ILS перед алгоритмом SA є його швидкодія. У частині графічних матеріалів наведені графіки часу роботи алгоритмів SA та ILS.

При однакових обмеженнях на час виконання для вісіх прогонів для алгоритму ILS булово витрачено на 35%менше часу. Слід зазначити, що з таблиці 5.2 видно, що з часом SA має тенденцію знаходити кращі розв’язки.

З огляду на специфіку задачі TOPTW можна виділити ряд переваг і недоліків реалізованих алгоритмів у порівнянні один з одним.

Перевагою модифікованого ILS є те, що через високу інтенсифікацію пошуку, час роботи алгоритму меншше. Він знаходить кращі розв’язки швидше за SA.

Недоліком ILS є той факт, що через присутність детермінованого критерію виходу і інтенсифікації імовірність потрапити у локальний оптимум в просторі розв’язків ЛП висока і подальші покращення розв’язку після цього неможливі.

Перевагою SA є те, що завдяки диверсифікації пошуку, з часом алгоритм здатен знаходити кращі розв’язки ніж модифікований ILS.

Недоліки SA:

– необхідно підбирати вхідні параметри, для регулювання температурного розкладу;

– необхідно генерувати початкову температуру, що потребує додаткового часу роботи, адже містить процедуру генерування ряду розв’язків;

– на загал довший час роботи.

5.3 Висновки до розділу

Проведено аналіз розроблених модифікацій алгоритму повторюваного локального пошуку і амітаційного відпалу. За результатами експериментальних досліджень встановлено, що запропоновані модифікації покращують швидкодію алгоритмів. Було порівняно алгоритми і наведено їх недоліки і переваги. Як наслідок, була досягнута мета досліджень.

ЗАГАЛЬНІ ВИСНОВКИ

В результаті огляду математичних постановок задачі побудови туристичних маршрутів предметом дослідження було обрано задачу TOPTW. Були наведені існуючі підходи до розв’язання TOPTW, описано практичні застосуваннння даної задачі.

Сформульована змістовна і математична постановка задачі; описано обрані методи розв’язання і їх специфіку відносно задачі TOPTW; створено алгоритмічне забезпечення для розв’язання задачі TOPTW на основі методів повторюваного локального пошуку та імітаційного відпалу; запропоновано модифікацію алгоритму повторюваного локального пошуку і покращено швидкодію використаних методів засобами паралельного програмування.

Було запрограмоване ПЗ яке реалізує алгоритмічне забезпечення для розв’язаннння TOPTW на мові C++ і користувацький інтерфейс на мові C#. Специфіка спроектованої архітектури і простий користувацький інтерфейс дають можливість легко розширювати функціональність.

Для дослідження ефективності розроблених алгоритмів було проведено експерименти і встановлено, що розроблено модифікації алгоритму повторюваного локального пошуку з виграшем у швидкодії.

Було вирішено всі задачі для досягнення поставленої мети роботи. В результаті роботи автоматизовано процес складання туристичних маршрутів, з умовою, що максимізується сумарна корисность побудованих туристичних маршрутів заданої тривалості з врахуванням часових періодів відвідування туристичних місць.

ПЕРЕЛІК ПОСИЛАНЬ

1. .NET [Електронний ресурс] – Режим доступу до ресурсу: https://www.microsoft.com/net/.

2. A matheuristic for the Team Orienteering Arc Routing Problem / [C. Archetti, A. Corbe, I. Plana та ін.]. // European Journal of Operational Research. – 2015. – №245. – С. 392–401.

3. A Matheuristic for the Team Orienteering Arc Routing Problem / [C. Archetti, A. Corberan, I. Plana та ін.]. // European Journal of Operational Research. – 2015. – №245. – С. 392–401.

4. A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery / [A. Subramanian, L. Drummond, C. Bentes та ін.]. // Computers & Operations Research. – 2010. – №37. – С. 1899–1911.

5. A survey on algorithmic approaches for solving tourist trip design problems / M.K, K. C, G. D, P. G. // Journal of Heuristics. – 2017

6. A guided local search metaheuristic for the team orienteering problem / P.Vansteenwegen, W. Souffriau, G. Berghe, D. Oudheusden. // European Journal of Operational Research. – 2009. – №196. – С. 118–127.

7. Aarts E. Simulated annealing / E. Aarts, J. Korst, P. Laarhoven // Local search in combinatorial optimization / E. Aarts, J. Korst, P. Laarhoven., 1997. – С. 91–120.

8. Alba E. Introduction to Metaheuristics and Parallelism / Enrique Alba // Parallel Metaheuristics: A New Class of Algorithms / Enrique Alba., 2005. – С. 3–42.

9. An adaptive ejection pool with toggle-rule diversification approach for the capacitated team orienteering problem / Z.Luo, B. Cheang, A. Lim, W. Zhu. // European Journal of Operational Research. – 2013. – №3. – С. 673 – 682.

10. Applegate D. Chained Lin-Kernighan for large traveling salesman problems / D. Applegate, A. Rohe, W. Cook. // INFORMS Journal on Computing. – 2003. – С. 82 – 92.

11. Araoz J. Privatized rural postman problems / J. Araoz, E. Fernandez, C. Zoltan. // Computers & Operations Research. – 2016. – №33. – С. 3432–3449.

12. Arkin E. Resource-constrained geometric network optimization / E. Arkin, J. Mitchell, G. Narasimhan. // 14th Annual Symposium on Computational Geometry. – 1998. – С. 307–316.

13. Baeza R. Modern Information Retrieval / R. Baeza , B. Ribeiro. – Нью-Йорк: Addison-Wesley, 1999. – 501 с.

14. Baum E. Iterated descent: A better algorithm for local search in combinatorial optimization problems / Baum // Technical report Caltech / Baum. – Pasadena, 1986.

15. Baum E. Towards practical “neural” computation for combinatorial optimization problems. / Baum. // AIP conference. – 1986. – С. 53–64.

16. Ben-Ameur W. Computing the Initial Temperature of Simulated Annealing / WALID Ben-Ameur. // Computational Optimization and Applications. – 2004. – №29. – С. 369–385.

17. Best-in-class tools for any developer [Електронний ресурс] – Режим доступу до ресурсу: https://www.visualstudio.com/.

18. Butt S. A heuristic for the multiple tour maximum collection problem / S. Butt, T. Cavalier. // Computers & Operations Research. – 1994. – №21. – С. 101 – 111.

19. C Sharp [Електронний ресурс] // Режим доступу: https://uk.wikipedia.org/wiki/C_Sharp.

20. C++ [Електронний ресурс] // Режим доступу: https://en.wikipedia.org/wiki/C%2B%2Bhttps://uk.wikipedia.org/wiki/C_Sharp.

21. Cerny V. A thermodynamical approach to the travelling salesman problem: An efficient simulation algorithm / Cerny. // Journal of Optimization Theory and Applications. – 1985. – №45. – С. 41–51.

22. Chardaire P. Thermostatistical persistency: A powerful improving concept for simulated annealing algorithms. / P. Chardaire, J. Lutton, A. Sutter. // European Journal of Operational Research. – 1995. – №86. – С. 565–579.

23. Chen L. A parallel ant colony algorithm on massively parallel processors and its convergence analysis for the travelling salesman problem / L. Chen, H. Sun, S. Wang. // Information Sciences. – 2012. – №199. – С. 31 – 42.

24. Common Language Runtime (CLR) [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/standard/clr.

25. Cordeau J. A parallel iterated tabu search heuristic for vehicle routing problems / J. Cordeau, M. Maischbergerb. // Computers & Operations Research. – 2012. – №39. – С. 2033–2050.

26. Crainic T. Parallel Solution Methods for Vehicle Routing Problems / Crainic // The Vehicle Routing Problem: Latest Advances and New Challenges / Crainic., 2008. – (Science and Business Media). – С. 497–542.

27. Dynamic-link library [Електронний ресурс] // Режим доступу: https://en.wikipedia.org/wiki/Dynamic-link_library.

28. Efficient Metaheuristics for the Mixed Team Orienteering Problem with Time Windows / [D. Gavalas, C. Konstantopoulos, K. Mastakas та ін.]. // Algorithms. – 2016. – №9. – С. 1–21.

29. Golden L. The orienteering problem / L. Golden, R. Vohra, L. Levy. // Naval Research Logistics. – 1987. – №34. – С. 307–318.

30. Gunawan A. Home Health Care Delivery Problem / A. Gunawan, L. Hoong, K. Kun. // Proceedings of the the 8th Multidisciplinary International Scheduling Conference. – 2017.

31. Hapsari I. Mobile Tourist Recommendation Systems Based On Tourist Trip Design Problem For Indonesia Domestic Tourist, An Exploratory Study / Indri Hapsari. // Widyatama International Seminar on Sustainability. – 2016. – №8.

32. Hertz A. Recent Trends in Arc Routing / Hertz. // Graph Theory, Combinatorics and Algorithms. – 2005. – №34. – С. 215–236.

33. Hu X. The team orienteering problem with capacity constraint and time window / X. Hu, Z. Li. // The 10th International Symposium on Operations Research and its Applications (ISORA 2011). – 2011. – С. 157–163.

34. Huawei. VRP Basics / Huawei // HCNA Networking Study Guide / Huawei., 2016. – С. 978–981.

35. Iterated local search for the team orienteering problem with time windows / P.Vansteenwegena, W. Souffriaua, G. Bergheb, D. Oudheusdena. // Computers & Operations Research. – 2009. – №36. – С. 3281–3290.

36. Johnson D. Local optimization and the travelling salesman problem / Johnson. // In: Proceedings of the 17th Colloquium on Automata, Languages, and Programming. Lecture Notes in Computer Science. – 1990. – №443. – С. 446–461.

37. Johnson D. The Traveling Salesman Problem: A Case Study in Local Optimization / D. Johnson, L. McGeoch // Local Search in Combinatorial Optimization / D. Johnson, L. McGeoch. – London: John Wiley and Sons, 1997. – С. 215–310.

38. Kataoka S. An algorithm for single constraint maximum collection problem / S. Kataoka, S. Morito. // Journal of the Operations Research Society of Japan. – 1988. – №31. – С. 515–530.

39. Kirkpatrick S. Optimization by simulated annealing / S. Kirkpatrick, C. Gelatt, M. Vecchi. // Science. – 1983. – С. 671–689.

40. Labadi N N. An Effective Hybrid Evolutionary Local Search for Orienteering and Team Orienteering Problems with Time Windows / N. Labadi N, J. Melechovsky, R. Calvo. // Parallel Problem Solving from Nature, PPSN XI. PPSN 2010. Lecture Notes in Computer Science. – 2010. – № 6239. – С. 219–228.

41. Labadi N N. Hybridized evolutionary local search algorithm for the team orienteering problem with time windows / N. Labadi N, J. Melechovsky, R. Calvo. // Journal of Heuristics. – 2011. – №17. – С. 729–753.

42. Laporte G. The selective travelling salesman problem / G. Laporte, S. Martello. // Discrete Applied Mathematics. – 1990. – №26. – С. 193 – 207.

43. Lin S. A simulated annealing heuristic for the team orienteering problem with time windows / S. Lin, V. Yu. // European Journal of Operational Research. – 2012. – №217. – С. 94 – 107..

44. Lourenço H. Iterated Local Search: Framework and Applications / H. Lourenço, O. Martin, T. Stützle // Handbook of Metaheuristics / H. Lourenço, O. Martin, T. Stützle., 2014. – (2). – (International Series in Operations Research & Management Science; кн. 146). – С. 363–397.

45. Lundy M. Convergence of an annealing algorithm / M. Lundy, A. Mees. // Mathematical Programming. – 1986. – №34. – С. 111–124.

46. Malandraki C. The maximum benefit Chinese postman problem and the maximum benefit traveling salesman problem / C. Malandraki, M. Daskin. // European Journal of Operational Research. – 1093. – №65. – С. 218–234.

47. Martin O. Combining simulated annealing with local search heuristics / O. Martin, S. Otto. // Ann. Oper. Res.. – 1996. – №63. – С. 57–75.

48. Martin O. Large-step Markov chains for the traveling salesman problem / O. Martin, S. Otto, E. Felten. // Complex Syst.. – 1991. – №5. – С. 299–326.

49. Microsoft Visual Studio Community 2017 [Електронний ресурс] – Режим доступу до ресурсу: https://www.visualstudio.com/license-terms/mlt553321/?rr=https%3A%2F%2Fwww.google.de%2F.

50. Montemanni R. An ant colony system for team orienteering problems with time windows / R. Montemanni, L. Gambardella. // Foundations of Computing and Decision Sciences. – 2009. – №34. – С. 287–306.

51. Multi-Agent task assignment for mobile crowdsourcing under trajectory uncertainties (extended abstract) / C.Chen, S. Cheng, A. Misra, H. Lau. // 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015). – 2015. – С. 4–8.

52. Niazi A. 10 advantages of C# programming language [Електронний ресурс] / ASAD Niazi – Режим доступу до ресурсу: http://proprogrammershub.blogspot.de/2016/04/top-10-advantages-of-c.html.

53. OP_FORMAT [Електронний ресурс] – Режим доступу до ресурсу: https://www.mech.kuleuven.be/en/cib/op/instances/OP_format/view.

54. OpenMp [Електронний ресурс] – Режим доступу до ресурсу: http://www.openmp.org/.

55. Osman I. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. / Osman. // Analysisof Operations Research. – 1993. – №41. – С. 421–451.

56. Pang K. An adaptive parallel route construction heuristic for the vehicle routing problem with time windows constraints / Pang. // Expert Systems with Applications. – 2011. – №38. – С. 11939–11946.

57. Pearn W. Transforming arc routing into node routing problems / W. Pearn, A. Assad, B. Golden. // Computers & Operations Research. – 1987. – №14. – С. 285–288.

58. Personalized Tourist Route Generation / [A. Garcia, O. Arbelaitz, M. Linaza та ін.]. // ICWE 2010: Current Trends in Web Engineering. – 2010. – №6385. – С. 486–497.

59. Souffriau W. A Personalised Tourist Trip Design Algorithmfor Mobile Tourist Guides / W. Souffriau, P. Vansteenwegen, J. Vertommen. – 2008.

60. Tackling large-scale home health care delivery problem with uncertainty. / C.Chen, Z. Rubinstein, S. Smith, H. Lau. // Twenty-Seventh International Conference on Automated Planning and Scheduling. – 2017. – №27. – С. 18–23.

61. Tae H. A Branch-and-Price Approach for the Team Orienteering Problem with Time Windows / H. Tae, B. Kim. // The International Journal of Industrial Engineering: Theory, Applications and Practice. – 2015. – №22. – С. 243–251.

62. The Chinese Postman Problem with Load-Dependent Costs / [A. Corberan, E. Gunes, G. Laporte та ін.]. // Transportation Science. – 2018. – №52. – С. 370 – 385.

63. The Orienteering Problem: Test Instances [Електронний ресурс] – Режим доступу до ресурсу: https://www.mech.kuleuven.be/en/cib/op.

64. The Stacker Crane Problem and the directed general routing problem / T.Avila, A. Corberan, I. Plana, J. Sanchis. // Networks - An International Journal. – 2015. – №56. – С. 43–55. Tsiligirides T. Heuristic methods applied to orienteering / Tsiligirides. // The Journal of the Operational Research Society.

65. The team orienteering problem with time windows: An lp-based granular variable neighborhood search / N.Labadi, J. Melechovsky, R. Woler Calvo, R. Mansini. // European Journal of Operational Research. – 2012. – №220. – С. 15–27.

66. Towards City-scale Mobile Crowdsourcing: Task Recommendations under Trajectory Uncertainties / C.Chen, S. Cheng, H. Lau, A. Misra. // International Joint Conference on Artificial Intelligence (IJCAI-2015). – 2015. – С. 25 – 31.

67. TRACCS: A Framework for Trajectory-Aware Coordinated Urban Crowd-Sourcing / [C. Chen, S. Cheng, A. Gunawan and oth.]. // HCOMP. – 2014.

68. Unmanaged Code [Електронний ресурс] – Режим доступу до ресурсу: https://msdn.microsoft.com/uk-ua/library/0e91td57(v=vs.100).aspx.

69. UNWTO [Електронний ресурс] – Режим доступу до ресурсу: http://www2.unwto.org/.

70. Vansteenwegen P. Planning in Tourism and Public Transportation - Attraction Selection by Means of a Personalised Electronic Tourist Guide and Train Transfer Scheduling / Vansteenwegen. // 4OR-Q J Oper Res. – 2009. – №61. – С. 7–293.

71. Vansteenwegen P. The mobile tourist guide: An or opportunity / P. Vansteenwegen, D. Van Oudheusden. // Operational Research Insight. – 2007. – №20. – С. 21–27.

72. Vansteenwegen P. The orienteering problem: A survey / P. Vansteenwegen, W. Soffuriau, D. Van Oudheusden. // European Journal of Operational Research. – 2011. – №209. – С. 1 – 10.

73. Visual Studio IDE overview [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/visualstudio/ide/visual-studio-ide.

74. Wasilc E. The team orienteering problem / E. Wasilc, B. Golden, I. MingChao. // European Journal of Operational Research. – 1996. – №88. – С. 464 – 474.

75. What Is Managed Code? [Електронний ресурс] // What Is Managed Code? – Режим доступу до ресурсу: https://msdn.microsoft.com/en-us/library/windows/desktop/bb318664(v=vs.85).aspx.

76. What's new in the .NET Framework [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/framework/whats-new/index.

77. White S. Concepts of scale in simulated annealing / White. // IEEE Int. Conference on Computer Design. – 1984.

78. Windows Forms [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/framework/winforms/.

79. Zenker B. ROSE - Assisting Pedestrians to Find Preferred Events and Comfortable Public Transport Connections / B. Zenker, B. Ludwig. // 6th International Conference on Mobile Technology, Application & Systems. – 2009. – С. 1–6.

80. Баран Г. Розвиток туризму в Україні: проблеми та перспективи [Електронний ресурс] / Г. Баран. – 2017. – Режим доступу до ресурсу: http://marker.ua/ua/sotsialnyj-blok/1804-razvitie-turizma-v-ukraine-problemy-i-perspektivy/.

81. Гуляницький Л. Ф. Прикладні методи комбінаторної оптимізації / Л. Ф. Гуляницький, О. Ю. Мулеса. – Київ: Видавничо-поліграфічний центр "Київський університет", 2016. – 142 с.

82. Прохорова К. Математична постановка та огляд методів розв’язування задачі побудови туристичних маршрутів / К. Прохорова, Л. Гуляницький. // Інформатика та обчислювальна техніка-IOT-2017. – 2017. – С. 78-89.

83. Прохорова К. С. Метод Розв’язання Задачі Командного Спортивного Орієнтування з Часовими Вікнами / К. С. Прохорова // Актуальні Питання Сьогодення / К. С. Прохорова. – Обухів: Типография «Друкарик», 2018. – (9). – С. 70–71.

84. Туризм в Україні: проблеми, перспективи розвитку [Електронний ресурс] – Режим доступу до ресурсу: http://skole.com.ua/uk/papers/13-turizm/36-turizmukrproblemi.html.

ДОДАТОК А ГРАФІЧНИЙ МАТЕРІАЛ

ПЛАКАТ 1 ЗМІСТОВНА ПОСТАНОВКА ЗАДАЧІ

ПЛАКАТ 2 БЛОК-СХЕМА ПОСЛІДОВНОГО АЛГОРИТМУ ПОВТОРЮВАНОГО ЛОКАЛЬНОГО ПОШУКУ

ПЛАКАТ 3 БЛОК-СХЕМА МОДИФІКОВАНОГО АЛГОРИТМУ ПОВТОРЮВАНОГО ЛОКАЛЬНОГО ПОШУКУ

ПЛАКАТ 4 СХЕМА СТРУКТУРНА КЛАСІВ БІБЛІОТЕКИ TOPTWLIB

ПЛАКАТ 5 РЕЗУЛЬТАТИ РОБОТИ МОДИФІКОВАНОГО ПОВТОРЮВАНОГО ЛОКАЛЬНОГО ПОШУКУ

ПЛАКАТ 6 ПОРІВНЯННЯ РЕЗУЛЬТАТІВ РОБОТИ ПАРАЛЕЛЬНОГО І ПОСЛІДОВНОГО АЛГОРИТМІВ

ПЛАКАТ 7 ЕКРАННІ ФОРМИ